Machine Learning Codes –

Chapter 1 (Weight Prediction)

```
!pip install --upgrade requests beautifulsoup4 pandas scikit-learn scipy
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (2.28.2)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.8/dist-packages (4.11.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages (1.5.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.8/dist-packages (1.2.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-packages (1.10.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests) (2022.12.7)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.8/dist-packages (from requests) (2.1.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests) (1.24.3)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.8/dist-packages (from beautifulsoup4) (2.3.2.post1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: numpy>=1.20.3 in /usr/local/lib/python3.8/dist-packages (from pandas) (1.21.6)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (3.1.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.8.1->pandas) (1.15.0)
```

Loading the Libraries

```
import math
import requests
from bs4 import BeautifulSoup
import pandas as pd
from sklearn import linear_model
from sklearn.metrics import mean_absolute_error
from scipy.stats import pearsonr
```

Understanding a Simple Weight Prediction Model

Loading and Preparing the Data

Load Webpage

```
[2] url = 'http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights'
    page = requests.get(url)
```

Extract Content

```
[3] soup = BeautifulSoup(page.content, 'html.parser')
    tbl = soup.find("table",{"class":"wikitable"})
```

+ Code

Convert to DataFrame

```
height_weight_df = pd.read_html(str(tbl))[0][['Height(Inches)','Weight(Pounds)']]
```

Count Records

```
[5]  num_records = height_weight_df.shape[0]
     print(num_records)
```

```
200
```

Place in x and y variables

```
[6]  x = height_weight_df['Height(Inches)'].values.reshape(num_records, 1)
     y = height_weight_df['Weight(Pounds)'].values.reshape(num_records, 1)
```

Fitting a Linear Regression Model

Fit Model

```
model = linear_model.LinearRegression()
_ = model.fit(x,y)
```

Generate Equation

```
[8]  print("ŷ = " + str(model.intercept_[0]) + " + " + str(model.coef_.T[0][0]) + " x₁")
```

```
ŷ = -106.02770644878137 + 3.4326761292716297 x₁
```

Compute Mean Absolute Error

```
[9]  y_pred = model.predict(x)
     mae = mean_absolute_error(y, y_pred)
     print(mae)
```

```
7.7587373803882205
```

Plot Regression Line ± Error

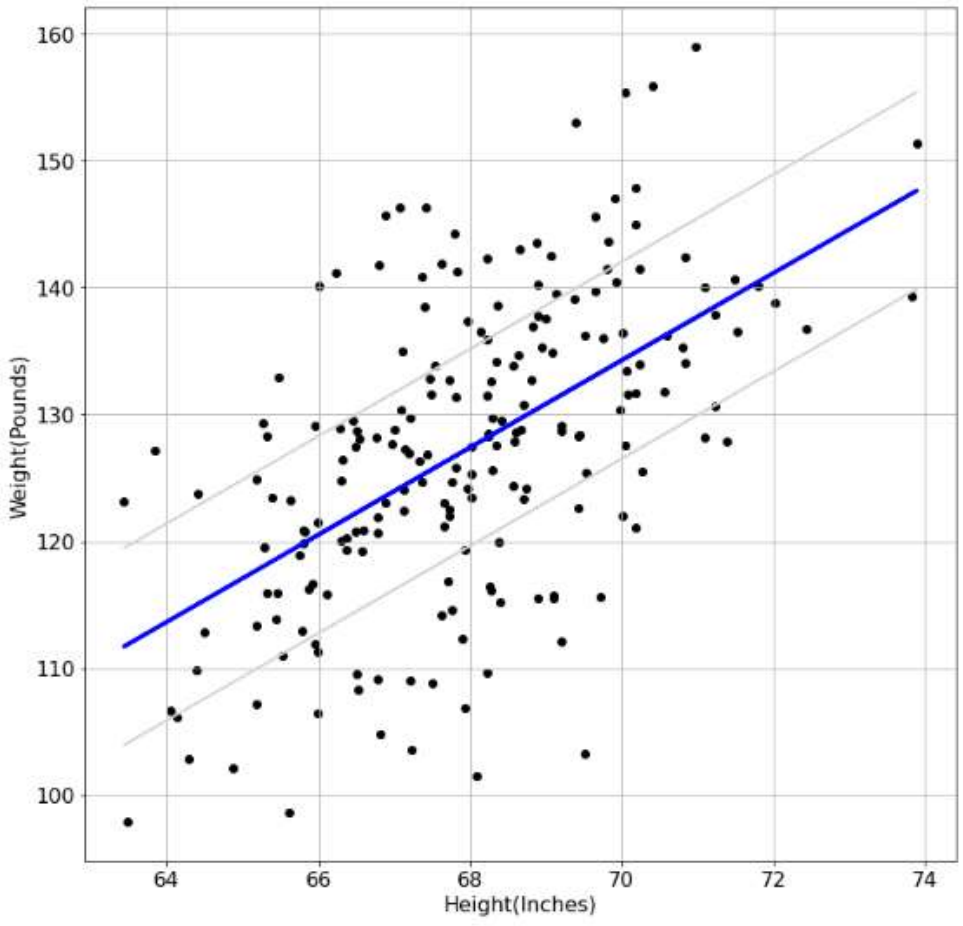
```
[10] pip install matplotlib==3.1.3
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting matplotlib==3.1.3
  Downloading matplotlib-3.1.3-cp38-cp38-manylinux1_x86_64.whl (13.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.1/13.1 MB 68.5 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.1.3) (1.21.6)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.1.3) (1.4.4)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.1.3) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.1.3) (0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.1.3) (3.0.9)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.1->matplotlib==3.1.3) (1.15.0)
Installing collected packages: matplotlib
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.2.2
    Uninstalling matplotlib-3.2.2:
      Successfully uninstalled matplotlib-3.2.2
Successfully installed matplotlib-3.1.3
WARNING: The following packages were previously imported in this runtime:
  [matplotlib,mpl_toolkits]
You must restart the runtime in order to use newly installed versions.
```

RESTART RUNTIME

```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(12,12))
plt.rcParams.update({'font.size': 16})
plt.scatter(x, y,  color='black')
plt.plot(x, y_pred, color='blue', linewidth=3)
plt.plot(x, y_pred + mae, color='lightgray')
plt.plot(x, y_pred - mae, color='lightgray')
plt.title('')
plt.xlabel('Height(Inches)')
plt.ylabel('Weight(Pounds)')
plt.grid(True)
plt.show()
```



Calculate Pearsons Correlation Coefficient

```
[13] corr, pval = pearsonr(x[:,0], y[:,0])
     print(corr)
```

```
0.5568647346122995
```

Two-tailed p-value

```
print(pval < 0.05)
```

```
True
```

Chapter 2 (CVD)

Installing the Libraries

These are all already installed on Google Colab by default so install only if running elsewhere (and ***not already installed***):

```
!pip install --upgrade pandas numpy statsmodels scikit-learn matplotlib
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages (1.3.5)
Collecting pandas
  Downloading pandas-1.5.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.2/12.2 MB 38.2 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (1.21.6)
Collecting numpy
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 66.1 MB/s eta 0:00:00
Requirement already satisfied: statsmodels in /usr/local/lib/python3.8/dist-packages (0.12.2)
Collecting statsmodels
  Downloading statsmodels-0.13.5-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (9.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.9/9.9 MB 50.9 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.8/dist-packages (1.0.2)
Collecting scikit-learn
  Downloading scikit_learn-1.2.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (9.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.8/9.8 MB 64.4 MB/s eta 0:00:00
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (3.2.2)
Collecting matplotlib
  Downloading matplotlib-3.6.3-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (9.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.4/9.4 MB 16.2 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: scipy>=1.3 in /usr/local/lib/python3.8/dist-packages (from statsmodels) (1.7.3)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.8/dist-packages (from statsmodels) (0.5.3)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.8/dist-packages (from statsmodels) (23.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.2.0)
```

Install these if running on Google Colab or ***not already installed***:

```
!pip install --upgrade machine-learning-datasets
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting machine-learning-datasets
  Downloading machine_learning_datasets-0.1.16.4-py3-none-any.whl (25 kB)
Requirement already satisfied: numpy<2.0.0,>=1.19.5 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (1.22.4)
Collecting pycebox<0.0.2,>=0.0.1
  Downloading pycebox-0.0.1.tar.gz (4.5 kB)
  Preparing metadata (setup.py) ... done
Collecting pathlib2<3.0.0,>=2.3.5
  Downloading pathlib2-2.3.7.post1-py2.py3-none-any.whl (18 kB)
Collecting aif360<0.4.0,>=0.3.0
  Downloading aif360-0.3.0-py3-none-any.whl (165 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 165.6/165.6 KB 17.8 MB/s eta 0:00:00
Requirement already satisfied: mlxtend<0.15.0,>=0.14.0 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (0.14.0)
Requirement already satisfied: matplotlib<4.0.0,>=3.2.2 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (3.6.3)
Requirement already satisfied: seaborn<0.12.0,>=0.11.1 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (0.11.2)
Collecting scikit-learn<0.23.0,>=0.22.2.post1
  Downloading scikit_learn-0.22.2.post1-cp38-cp38-manylinux1_x86_64.whl (7.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.0/7.0 MB 81.9 MB/s eta 0:00:00
Requirement already satisfied: pandas<2.0.0,>=1.1.5 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (1.5.3)
Collecting statsmodels<0.11.0,>=0.10.2
  Downloading statsmodels-0.10.2-cp38-cp38-manylinux1_x86_64.whl (8.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.1/8.1 MB 70.0 MB/s eta 0:00:00
Requirement already satisfied: scipy<2.0.0,>=1.4.1 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (1.7.3)
Requirement already satisfied: tqdm<5.0.0,>=4.41.1 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (4.64.1)
Collecting alibi<0.6.0,>=0.5.5
  Downloading alibi-0.5.8-py3-none-any.whl (312 kB)
```

Loading the Libraries

```
import math
import machine_learning_datasets as mldatasets
import pandas as pd
import numpy as np
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

```
/usr/local/lib/python3.8/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util
  import pandas.util.testing as tm
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/base/tsa_model.py:8: FutureWarning: pandas.I
  from pandas import (to_datetime, Int64Index, DatetimeIndex, Period,
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/base/tsa_model.py:8: FutureWarning: pandas.F
  from pandas import (to_datetime, Int64Index, DatetimeIndex, Period,
```

```
cvd_df = mldatasets.load("cardiovascular-disease")
```

```
https://raw.githubusercontent.com/caravanuden/cardio/master/cardio_train.csv downloaded to /content/data/cardio_train.csv
1 dataset files found in /content/data folder
parsing /content/data/cardio_train.csv
```

```
cvd_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   age          70000 non-null  int64  
 1   gender       70000 non-null  int64  
 2   height       70000 non-null  int64  
 3   weight       70000 non-null  float64
 4   ap_hi        70000 non-null  int64  
 5   ap_lo        70000 non-null  int64  
 6   cholesterol  70000 non-null  int64  
 7   gluc         70000 non-null  int64  
 8   smoke        70000 non-null  int64  
 9   alco         70000 non-null  int64  
 10  active       70000 non-null  int64  
 11  cardio       70000 non-null  int64  
dtypes: float64(1), int64(11)
memory usage: 6.4 MB
```

```
cvd_df['age'] =  cvd_df['age'] / 365.24
```

+ Code + Text

```
cvd_df.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
age	70000.0	53.304309	6.755152	29.564122	48.36272	53.945351	58.391742	64.924433
gender	70000.0	1.349571	0.476838	1.000000	1.00000	1.000000	2.000000	2.000000
height	70000.0	164.359229	8.210126	55.000000	159.00000	165.000000	170.000000	250.000000
weight	70000.0	74.205690	14.395757	10.000000	65.00000	72.000000	82.000000	200.000000
ap_hi	70000.0	128.817286	154.011419	-150.000000	120.00000	120.000000	140.000000	16020.000000
ap_lo	70000.0	96.630414	188.472530	-70.000000	80.00000	80.000000	90.000000	11000.000000
cholesterol	70000.0	1.366871	0.680250	1.000000	1.00000	1.000000	2.000000	3.000000
gluc	70000.0	1.226457	0.572270	1.000000	1.00000	1.000000	1.000000	3.000000
smoke	70000.0	0.088129	0.283484	0.000000	0.00000	0.000000	0.000000	1.000000
alco	70000.0	0.053771	0.225568	0.000000	0.00000	0.000000	0.000000	1.000000
active	70000.0	0.803729	0.397179	0.000000	1.00000	1.000000	1.000000	1.000000
cardio	70000.0	0.499700	0.500003	0.000000	0.00000	0.000000	1.000000	1.000000

```
cvd_df = cvd_df[(cvd_df['ap_lo'] <= 370) & (cvd_df['ap_lo'] > 0)].reset_index(drop=True)
cvd_df = cvd_df[(cvd_df['ap_hi'] <= 370) & (cvd_df['ap_hi'] > 0)].reset_index(drop=True)
cvd_df = cvd_df[cvd_df['ap_hi'] >= cvd_df['ap_lo']].reset_index(drop=True)
```

```
y = cvd_df['cardio']
X = cvd_df.drop(['cardio'], axis=1).copy()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=9)
```

+ Code + Text

```
log_model = sm.Logit(y_train, sm.add_constant(X_train))
log_result = log_model.fit()
print(log_result.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.561557
         Iterations 6
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.190
Dependent Variable: cardio           AIC:              65618.3485
Date:               2023-02-06 17:33 BIC:              65726.0502
No. Observations:   58404            Log-Likelihood:   -32797.
Df Model:           11               LL-Null:          -40481.
Df Residuals:       58392            LLR p-value:      0.0000
Converged:          1.0000           Scale:            1.0000
No. Iterations:     6.0000
-----------------------------------------------------------------
               Coef.   Std.Err.     z     P>|z|   [0.025   0.975]
-----------------------------------------------------------------
const        -11.1730   0.2504 -44.6182 0.0000 -11.6638 -10.6822
age            0.0510   0.0015  34.7971 0.0000   0.0482   0.0539
gender        -0.0227   0.0238  -0.9568 0.3387  -0.0693   0.0238
height        -0.0036   0.0014  -2.6028 0.0092  -0.0063  -0.0009
weight         0.0111   0.0007  14.8567 0.0000   0.0096   0.0125
ap_hi          0.0561   0.0010  56.2824 0.0000   0.0541   0.0580
ap_lo          0.0105   0.0016   6.7670 0.0000   0.0075   0.0136
cholesterol    0.4931   0.0169  29.1612 0.0000   0.4600   0.5262
gluc          -0.1155   0.0192  -6.0138 0.0000  -0.1532  -0.0779
smoke         -0.1306   0.0376  -3.4717 0.0005  -0.2043  -0.0569
alco          -0.2050   0.0457  -4.4907 0.0000  -0.2945  -0.1155
active        -0.2151   0.0237  -9.0574 0.0000  -0.2616  -0.1685
-----------------------------------------------------------------
```

```
np.exp(log_result.params).sort_values(ascending=False)
```

```
cholesterol    1.637374
ap_hi          1.057676
age            1.052357
weight         1.011129
ap_lo          1.010573
height         0.996389
gender         0.977519
gluc           0.890913
smoke          0.877576
alco           0.814627
active         0.806471
const          0.000014
dtype: float64
```

```
np.std(X_train, 0)
```

```
age             6.757537
gender          0.476697
height          8.186987
weight         14.335173
ap_hi          16.703572
ap_lo           9.547583
cholesterol     0.678878
gluc            0.571231
smoke           0.283629
alco            0.225483
active          0.397215
dtype: float64
```

```
coefs = log_result.params.drop(labels=['const','gender'])
stdv = np.std(X_train, 0).drop(labels='gender')
abs(coefs * stdv).sort_values(ascending=False)
```

```
ap_hi          0.936632
age            0.344855
cholesterol    0.334750
weight         0.158651
ap_lo          0.100419
active         0.085436
gluc           0.065982
alco           0.046230
smoke          0.037040
height         0.029620
dtype: float64
```

```
print(y_pred[2872])
```

```
0.5746680418975686
```

```
print(X_test.iloc[2872])
```

```
age             60.521849
gender           1.000000
height         158.000000
weight          62.000000
ap_hi          130.000000
ap_lo           80.000000
cholesterol      1.000000
gluc             1.000000
smoke            0.000000
alco             0.000000
active           1.000000
Name: 46965, dtype: float64
```

```
filler_feature_values = {0: 1, 1: 30, 2: 1, 3: 165, 4: 57, 5: 110,
                         6: 70, 7: 1, 8: 1, 9: 0, 10:0, 11:1
                        }
```

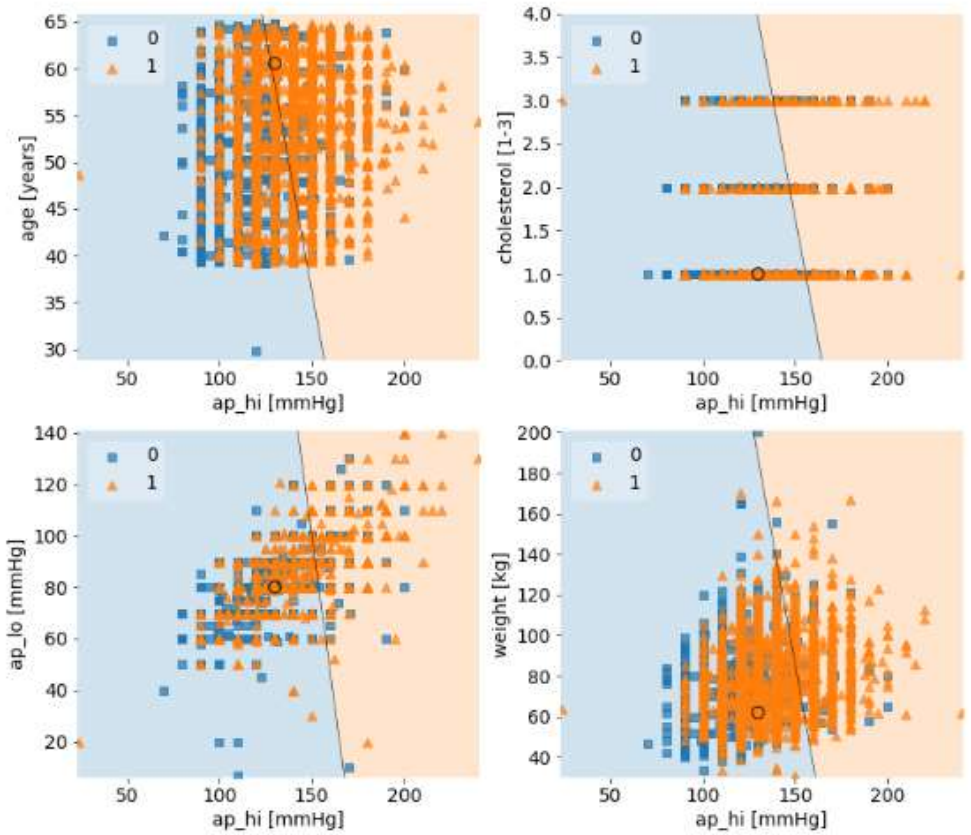
```
filler_feature_ranges = {0: 1, 1: 35, 2: 2, 3: 110, 4: 150, 5: 140,
                         6: 70, 7: 3, 8: 3, 9: 2, 10:2, 11:2
                        }
```

```
X_highlight = np.reshape(np.concatenate(([1],X_test.iloc[2872].to_numpy())), (1, 12))
print(X_highlight)
```

```
[[  1.          60.52184865   1.         158.          62.
  130.          80.           1.           1.           0.
    0.           1.        ]]
```

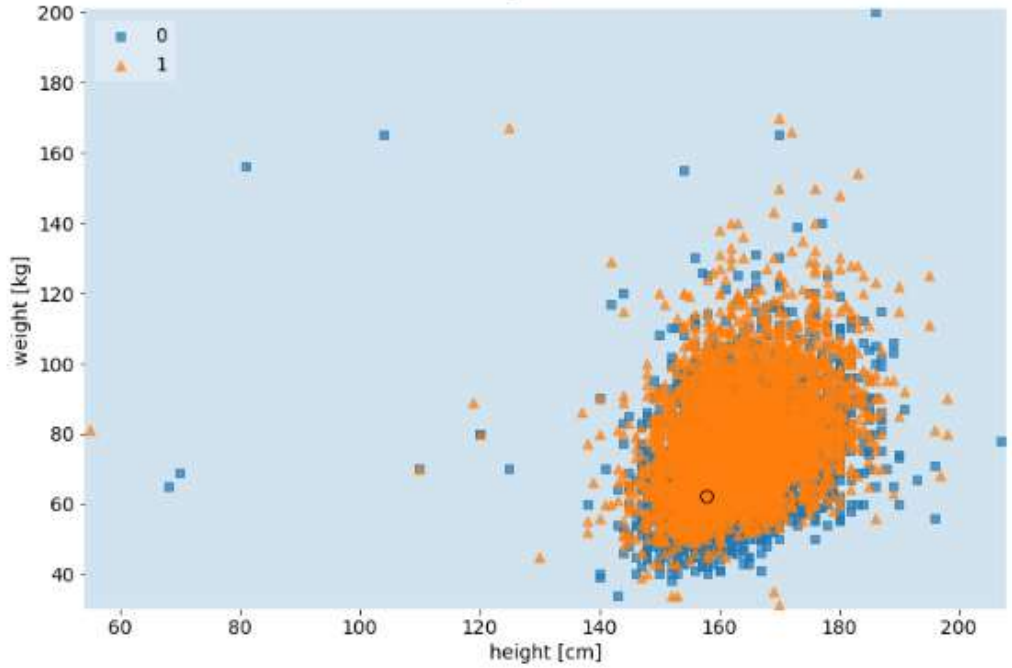
```
pip install matplotlib==3.1.3

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting matplotlib==3.1.3
  Downloading matplotlib-3.1.3-cp38-cp38-manylinux1_x86_64.whl (13.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.1/13.1 MB 45.8 MB/s eta 0:00:00
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.1.3) (0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.1.3) (3.0.9)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.1.3) (1.4.4)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.1.3) (1.22.4)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.1.3) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.1->matplotlib==3.1.3) (1.15.0)
Installing collected packages: matplotlib
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.5.0
    Uninstalling matplotlib-3.5.0:
      Successfully uninstalled matplotlib-3.5.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
yellowbrick 1.5 requires scikit-learn>=1.0.0, but you have scikit-learn 0.22.2.post1 which is incompatible.
plotnine 0.8.0 requires scipy>=1.5.0, but you have scipy 1.4.1 which is incompatible.
plotnine 0.8.0 requires statsmodels>=0.12.1, but you have statsmodels 0.10.2 which is incompatible.
machine-learning-datasets 0.1.16.4 requires matplotlib<4.0.0,>=3.2.2, but you have matplotlib 3.1.3 which is incompatible.
Successfully installed matplotlib-3.1.3
```

```
plt.rcParams.update({'font.size': 14})
fig, axarr = plt.subplots(2, 2, figsize=(12,8), sharex=True, sharey=False)
mldatasets.create_decision_plot(X_test, y_test, log_result, [5, 1], ['ap_hi [mmHg]', 'age [years]'],
                    X_highlight, filler_feature_values, filler_feature_ranges, ax=axarr.flat[0])
mldatasets.create_decision_plot(X_test, y_test, log_result, [5, 7], ['ap_hi [mmHg]', 'cholesterol [1-3]'],
                    X_highlight, filler_feature_values, filler_feature_ranges, ax=axarr.flat[1])
mldatasets.create_decision_plot(X_test, y_test, log_result, [5, 6], ['ap_hi [mmHg]', 'ap_lo [mmHg]'],
                    X_highlight, filler_feature_values, filler_feature_ranges, ax=axarr.flat[2])
mldatasets.create_decision_plot(X_test, y_test, log_result, [5, 4], ['ap_hi [mmHg]', 'weight [kg]'],
                    X_highlight, filler_feature_values, filler_feature_ranges, ax=axarr.flat[3])
plt.subplots_adjust(top = 1, bottom=0, hspace=0.2, wspace=0.2)
plt.show()
```



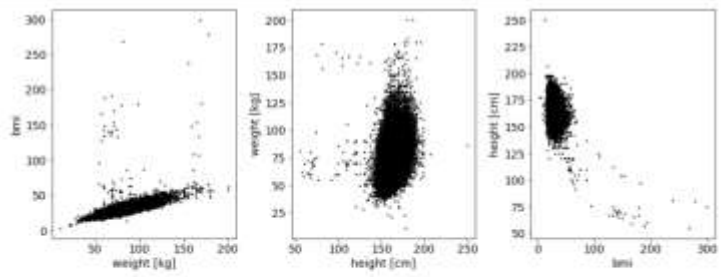
```
fig, ax = plt.subplots(1,1, figsize=(12,8))
mldatasets.create_decision_plot(X_test, y_test, log_result, [3, 4], ['height [cm]', 'weight [kg]'],
                    X_highlight, filler_feature_values, filler_feature_ranges, ax=ax)
plt.show()
```

```
No contour levels were found within the data range.
```




```
X2 = cvd_df.drop(['cardio'], axis=1).copy()
X2["bmi"] = X2["weight"] / (X2["height"]/100)**2
```

```
fig, axs = plt.subplots(1,3, figsize=(15,4))
plt.rcParams.update({'font.size': 14})
axs[0].scatter(X2["weight"], X2["bmi"], color='black', s=2)
axs[0].set_xlabel('weight [kg]')
axs[0].set_ylabel('bmi')
axs[1].scatter(X2["height"], X2["weight"], color='black', s=2)
axs[1].set_xlabel('height [cm]')
axs[1].set_ylabel('weight [kg]')
axs[2].scatter(X2["bmi"], X2["height"], color='black', s=2)
axs[2].set_xlabel('bmi')
axs[2].set_ylabel('height [cm]')
plt.subplots_adjust(top = 1, bottom=0, hspace=0.2, wspace=0.3)
plt.show()
```




```
X2 = X2.drop(['weight','height'], axis=1)
X2_train, X2_test, _, _ = train_test_split(X2, y, test_size=0.15, random_state=9)
```

```
log_model2 = sm.Logit(y_train, sm.add_constant(X2_train))
log_result2 = log_model2.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.562104
         Iterations 6
```

```
filler_feature_values2 = {0: 1, 1: 60, 2: 1, 3: 110,
                          4: 70, 5: 1, 6: 1, 7: 0, 8:0, 9:1, 10:20
                         }
filler_feature_ranges2 = {0: 1, 1: 35, 2: 2, 3: 140,
                          4: 70, 5: 3, 6: 3, 7: 2, 8:2, 9:2, 10:250
                         }
X2_highlight = np.reshape(np.concatenate(([1],X2_test.iloc[2872].to_numpy())), (1, 11))
fig, ax = plt.subplots(1,1, figsize=(12,8))
mldatasets.create_decision_plot(X2_test, y_test, log_result2, [3, 10], ['ap_hi [mmHg]', 'bmi'],
                    X2_highlight, filler_feature_values2, filler_feature_ranges2, ax=ax)
plt.show()
```

Chapter 3 (Flight Delay)

```
!pip install --upgrade machine-learning-datasets
!pip install --upgrade rulefit interpret skope-rules
!pip install --no-deps git+https://github.com/maxfrenzel/CompressionVAE.git
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting machine-learning-datasets
  Downloading machine_learning_datasets-0.1.16.4-py3-none-any.whl (25 kB)
Requirement already satisfied: opencv-python<5.0.0,>=4.5.1 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (4.6.0.66)
Collecting alibi<0.6.0,>=0.5.5
  Downloading alibi-0.5.8-py3-none-any.whl (312 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 312.5/312.5 KB 15.5 MB/s eta 0:00:00
Collecting seaborn<0.12.0,>=0.11.1
  Downloading seaborn-0.11.2-py3-none-any.whl (292 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 292.8/292.8 KB 26.2 MB/s eta 0:00:00
Collecting scikit-learn<0.25.0,>=0.22.2.post1
  Downloading scikit_learn-0.22.2.post1-cp38-cp38-manylinux1_x86_64.whl (7.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.0/7.0 MB 64.1 MB/s eta 0:00:00
Collecting pycebox<0.0.2,>=0.0.1
  Downloading pycebox-0.0.1.tar.gz (4.5 kB)
  Preparing metadata (setup.py) ... done
Collecting statsmodels<0.11.0,>=0.10.2
  Downloading statsmodels-0.10.2-cp38-cp38-manylinux1_x86_64.whl (8.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.1/8.1 MB 69.1 MB/s eta 0:00:00
Collecting pathlib2<3.0.0,>=2.3.5
  Downloading pathlib2-2.3.7.post1-py2.py3-none-any.whl (18 kB)
Requirement already satisfied: pandas<2.0.0,>=1.1.5 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (1.5.3)
Requirement already satisfied: matplotlib<4.0.0,>=3.2.2 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (3.6.3)
Requirement already satisfied: tqdm<5.0.0,>=4.41.1 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (4.64.1)
Collecting aif360<0.4.0,>=0.3.0
  Downloading aif360-0.3.0-py3-none-any.whl (165 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 165.6/165.6 KB 18.7 MB/s eta 0:00:00
```

```
import math
import machine_learning_datasets as mldatasets
import pandas as pd
import numpy as np
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import metrics, linear_model, tree, naive_bayes, neighbors, ensemble,\
                    neural_network, svm, decomposition, manifold
from rulefit import RuleFit
import statsmodels.api as sm
from interpret.glassbox import ExplainableBoostingClassifier
from interpret import show
from interpret.perf import ROC
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import sys
#Next two lines of code only needed while CVAE
#remains incompatible to Tensorflow 2.2+
import tensorflow.compat.v1 as tf
sys.modules['tensorflow'] = tf
from cvae import cvae
#Next two lines of code only needed while SkopedRules
#remains incompatible to Sklearn 0.23.0+
import six
sys.modules['sklearn.externals.six'] = six
from skrules import SkopeRules
```

```
audio_df = mldatasets.load("aa-domestic-delays-2018")
```

```
https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python/raw/master/datasets/aa-domestic-delays-2018.csv.zip downloaded to /Users/serg/EXTEND/PROJECTS/InterpretableMLBook/p
/Users/serg/EXTEND/PROJECTS/InterpretableMLBook/programming/Chapter3/data/aa-domestic-delays-2018.csv.zip uncompressed to /Users/serg/EXTEND/PROJECTS/InterpretableMLBook/programming/Chapter3/da
1 dataset files found in /Users/serg/EXTEND/PROJECTS/InterpretableMLBook/programming/Chapter3/data/aa-domestic-delays-2018.csv folder
parsing /Users/serg/EXTEND/PROJECTS/InterpretableMLBook/programming/Chapter3/data/aa-domestic-delays-2018.csv/aa-domestic-delays-2018.csv
```

```
aad18_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 899527 entries, 0 to 899526
Data columns (total 23 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   FL_NUM                 899527 non-null  int64  
 1   ORIGIN                 899527 non-null  object 
 2   DEST                   899527 non-null  object 
 3   PLANNED_DEP_DATETIME   899527 non-null  object 
 4   CRS_DEP_TIME           899527 non-null  int64  
 5   DEP_TIME               899527 non-null  float64
 6   DEP_DELAY              899527 non-null  float64
 7   DEP_AFPH               899527 non-null  float64
 8   DEP_RFPH               899527 non-null  float64
 9   TAXI_OUT               899527 non-null  float64
 10  WHEELS_OFF             899527 non-null  float64
 11  CRS_ELAPSED_TIME       899527 non-null  float64
 12  PCT_ELAPSED_TIME       899527 non-null  float64
 13  DISTANCE               899527 non-null  float64
 14  CRS_ARR_TIME           899527 non-null  int64  
 15  ARR_AFPH               899527 non-null  float64
 16  ARR_RFPH               899527 non-null  float64
 17  ARR_DELAY              899527 non-null  float64
 18  CARRIER_DELAY          899527 non-null  float64
 19  WEATHER_DELAY          899527 non-null  float64
 20  NAS_DELAY              899527 non-null  float64
 21  SECURITY_DELAY         899527 non-null  float64
 22  LATE_AIRCRAFT_DELAY    899527 non-null  float64
dtypes: float64(17), int64(3), object(3)
memory usage: 157.8+ MB
```

```
aad18_df['PLANNED_DEP_DATETIME'] = pd.to_datetime(aad18_df['PLANNED_DEP_DATETIME'])
```

```
aad18_df['DEP_MONTH'] = aad18_df['PLANNED_DEP_DATETIME'].dt.month
aad18_df['DEP_DOW'] = aad18_df['PLANNED_DEP_DATETIME'].dt.dayofweek
```

```
aad18_df = aad18_df.drop(['PLANNED_DEP_DATETIME'], axis=1)
```

```
#Create list with 10 hubs
hubs = ['CLT', 'ORD', 'DFW', 'LAX', 'MIA', 'JFK', 'LGA', 'PHL', 'PHX', 'DCA']
#Boolean series for if ORIGIN or DEST are hubs
is_origin_hub = aad18_df['ORIGIN'].isin(hubs)
is_dest_hub = aad18_df['DEST'].isin(hubs)
#Use boolean series to set ORIGIN_HUB and DEST_HUB
aad18_df['ORIGIN_HUB'] = 0
aad18_df.loc[is_origin_hub, 'ORIGIN_HUB'] = 1
aad18_df['DEST_HUB'] = 0
aad18_df.loc[is_dest_hub, 'DEST_HUB'] = 1
#Delete columns with codes
aad18_df = aad18_df.drop(['FL_NUM', 'ORIGIN', 'DEST'], axis=1)
```

```
aad18_df.loc[aad18_df['ARR_DELAY'] > 15, ['ARR_DELAY','CARRIER_DELAY']].head(10)
```

	ARR_DELAY	CARRIER_DELAY
8	168.0	136.0
16	20.0	5.0
18	242.0	242.0
19	62.0	62.0
22	19.0	19.0
26	26.0	0.0
29	77.0	77.0
32	19.0	19.0
33	18.0	1.0
40	36.0	16.0

```
aad18_df = aad18_df.drop(['ARR_DELAY'], axis=1)
```

```
rand = 9
y = aad18_df['CARRIER_DELAY']
X = aad18_df.drop(['CARRIER_DELAY'], axis=1).copy()
X_train, X_test, y_train_reg, y_test_reg = train_test_split(X, y, test_size=0.15, random_state=rand)
y_train_class = y_train_reg.apply(lambda x: 1 if x > 15 else 0)
y_test_class = y_test_reg.apply(lambda x: 1 if x > 15 else 0)
```

+ Code + Text

```
corr = aad18_df.corr()
abs(corr['CARRIER_DELAY']).sort_values(ascending=False)
```

```
CARRIER_DELAY          1.000000
DEP_DELAY              0.703935
ARR_RFPH               0.101742
LATE_AIRCRAFT_DELAY    0.083166
DEP_RFPH               0.058659
ARR_AFPH               0.035135
DEP_TIME               0.030941
NAS_DELAY              0.026792
WHEELS_OFF             0.026787
TAXI_OUT               0.024635
PCT_ELAPSED_TIME       0.020980
CRS_DEP_TIME           0.016032
ORIGIN_HUB             0.015334
DEST_HUB               0.013932
DISTANCE               0.010680
DEP_MONTH              0.009728
CRS_ELAPSED_TIME       0.008801
DEP_DOW                0.007043
CRS_ARR_TIME           0.007029
DEP_AFPH               0.006053
WEATHER_DELAY          0.003002
SECURITY_DELAY         0.000460
Name: CARRIER_DELAY, dtype: float64
```

Evaluating by Regression Models

```
reg_models = {
        #Generalized Linear Models (GLMs)
        'linear':{'model': linear_model.LinearRegression()},
        'linear_poly':{'model': make_pipeline(PolynomialFeatures(degree=2),
                                linear_model.LinearRegression(fit_intercept=False)) },
        'linear_interact':{'model': make_pipeline(PolynomialFeatures(interaction_only=True),
                                linear_model.LinearRegression(fit_intercept=False)) },
        'ridge':{'model': linear_model.RidgeCV(alphas=[1e-3, 1e-2, 1e-1, 1]) },
        #Trees
        'decision_tree':{'model': tree.DecisionTreeRegressor(max_depth=7, random_state=rand)},
        #RuleFit
        'rulefit':{'model': RuleFit(max_rules=150, rfmode='regress', random_state=rand)},
        #Nearest Neighbors
        'knn':{'model': neighbors.KNeighborsRegressor(n_neighbors=7)},
        #Ensemble Methods
        'random_forest':{'model':ensemble.RandomForestRegressor(max_depth=7, random_state=rand)},
        #Neural Networks
        'mlp':{'model':neural_network.MLPRegressor(hidden_layer_sizes=(21,), max_iter=500,
                                                   early_stopping=True, random_state=rand)}
    }
```

```
for model_name in reg_models.keys():
    if model_name != 'rulefit':
        fitted_model = reg_models[model_name]['model'].fit(X_train, y_train_reg)
    else:
        fitted_model = reg_models[model_name]['model'].fit(X_train.values, y_train_reg.values, X_test.columns)
    y_train_pred = fitted_model.predict(X_train.values)
    y_test_pred = fitted_model.predict(X_test.values)
    reg_models[model_name]['fitted'] = fitted_model
    reg_models[model_name]['preds'] = y_test_pred
    reg_models[model_name]['RMSE_train'] = math.sqrt(metrics.mean_squared_error(y_train_reg, y_train_pred))
    reg_models[model_name]['RMSE_test'] = math.sqrt(metrics.mean_squared_error(y_test_reg, y_test_pred))
    reg_models[model_name]['R2_test'] = metrics.r2_score(y_test_reg, y_test_pred)
```

```
reg_metrics = pd.DataFrame.from_dict(reg_models, 'index')[['RMSE_train', 'RMSE_test', 'R2_test']]
reg_metrics.sort_values(by='RMSE_test').style.\
    background_gradient(cmap='viridis', low=1, high=0.3, subset=['RMSE_train', 'RMSE_test']).\
    background_gradient(cmap='plasma', low=0.3, high=1, subset=['R2_test'])
```


```
reg_metrics = pd.DataFrame.from_dict(reg_models, 'index')[['RMSE_train', 'RMSE_test', 'R2_test']]
reg_metrics.sort_values(by='RMSE_test').style.\
    background_gradient(cmap='viridis', low=1, high=0.3, subset=['RMSE_train', 'RMSE_test']).\
    background_gradient(cmap='plasma', low=0.3, high=1, subset=['R2_test'])
```


	RMSE_train	RMSE_test	R2_test
mlp	3.243516	3.308597	0.987025
random_forest	5.143267	6.088249	0.956065
linear_poly	6.213987	6.339854	0.952359
linear_interact	6.454305	6.562285	0.948957
decision_tree	6.542924	7.456335	0.934102
linear	7.819643	7.882875	0.926347
ridge	8.122859	8.168530	0.920912
knn	7.360098	9.259422	0.898377
rulefit	9.171494	9.308878	0.897289

Evaluating by Classification Models

```
class_models = {
        #Generalized Linear Models (GLMs)
        'logistic':{'model': linear_model.LogisticRegression()},
        'ridge':{'model': linear_model.RidgeClassifierCV(cv=5, alphas=[1e-3, 1e-2, 1e-1, 1],\
                                                         class_weight='balanced')},
        #Tree
        'decision_tree':{'model': tree.DecisionTreeClassifier(max_depth=7, random_state=rand)},
        #Nearest Neighbors
        'knn':{'model': neighbors.KNeighborsClassifier(n_neighbors=7)},
        #Naive Bayes
        'naive_bayes':{'model': naive_bayes.GaussianNB()},
        #Ensemble Methods
        'gradient_boosting':{'model':ensemble.GradientBoostingClassifier(n_estimators=210)},
        'random_forest':{'model':ensemble.RandomForestClassifier(max_depth=11,\
                                                                 class_weight='balanced', random_state=rand)},
        #Neural Networks
        'mlp':{'model':make_pipeline(StandardScaler(),\
                                     neural_network.MLPClassifier(hidden_layer_sizes=(7,), max_iter=500,\
                                                                  early_stopping=True, random_state=rand))}
    }
```

```
print(y_train_class[y_train_class==1].shape[0] / y_train_class.shape[0])
```

```
0.061283264255549
```

```
for model_name in class_models.keys():
    fitted_model = class_models[model_name]['model'].fit(X_train, y_train_class)
    y_train_pred = fitted_model.predict(X_train.values)
    if model_name == 'ridge':
        y_test_pred = fitted_model.predict(X_test.values)
    else:
        y_test_prob = fitted_model.predict_proba(X_test.values)[:,1]
        y_test_pred = np.where(y_test_prob > 0.5, 1, 0)
    class_models[model_name]['fitted'] = fitted_model
    class_models[model_name]['probs'] = y_test_prob
    class_models[model_name]['preds'] = y_test_pred
    class_models[model_name]['Accuracy_train'] = metrics.accuracy_score(y_train_class, y_train_pred)
    class_models[model_name]['Accuracy_test'] = metrics.accuracy_score(y_test_class, y_test_pred)
    class_models[model_name]['Recall_train'] = metrics.recall_score(y_train_class, y_train_pred)
    class_models[model_name]['Recall_test'] = metrics.recall_score(y_test_class, y_test_pred)
    if model_name != 'ridge':
        class_models[model_name]['ROC_AUC_test'] = metrics.roc_auc_score(y_test_class, y_test_prob)
    else:
        class_models[model_name]['ROC_AUC_test'] = 0
    class_models[model_name]['F1_test'] = metrics.f1_score(y_test_class, y_test_pred)
    class_models[model_name]['MCC_test'] = metrics.matthews_corrcoef(y_test_class, y_test_pred)
```

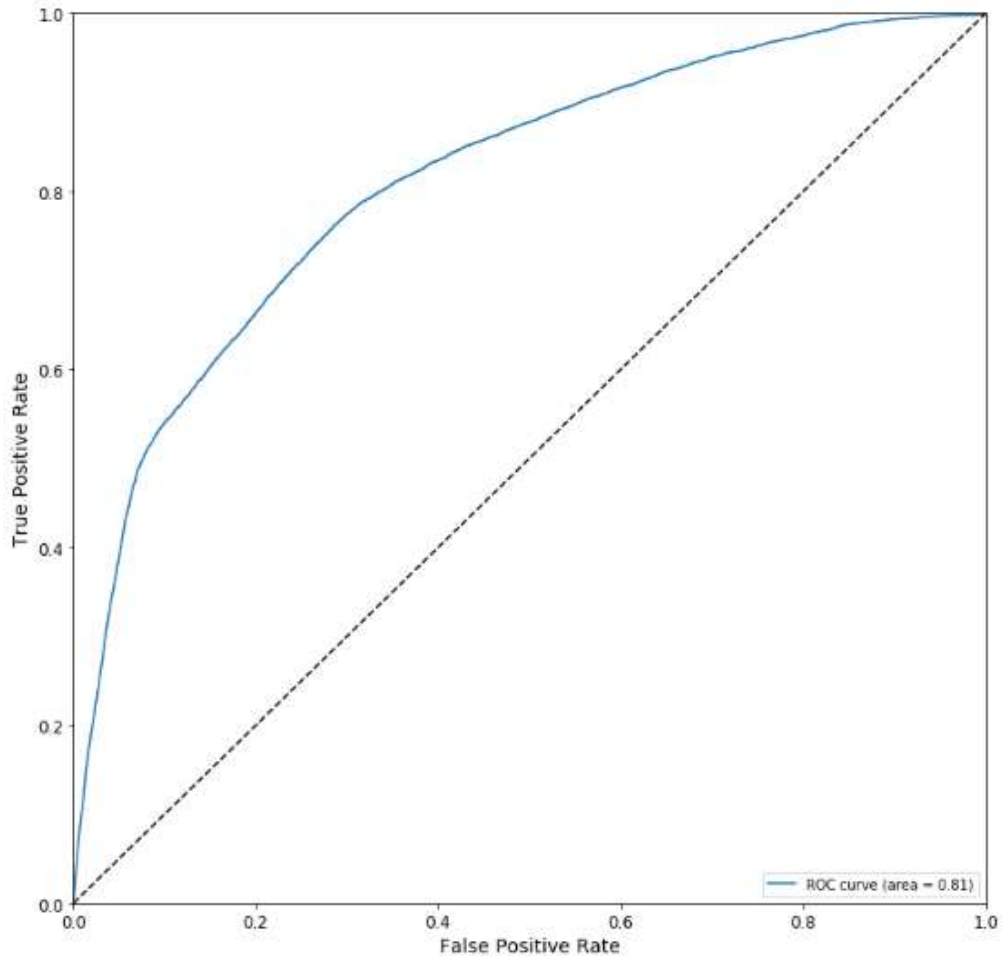
```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

```
class_metrics = pd.DataFrame.from_dict(class_models, 'index')[['Accuracy_train', 'Accuracy_test',\
                                                               'Recall_train', 'Recall_test',\
                                                               'ROC_AUC_test', 'F1_test', 'MCC_test']]
class_metrics.sort_values(by='ROC_AUC_test', ascending=False).style.\
    background_gradient(cmap='plasma', low=0.3, high=1, subset=['Accuracy_train', 'Accuracy_test']).\
    background_gradient(cmap='viridis', low=1, high=0.3, subset=['Recall_train', 'Recall_test',\
                                                                 'ROC_AUC_test', 'F1_test', 'MCC_test'])
```

	Accuracy_train	Accuracy_test	Recall_train	Recall_test	ROC_AUC_test	F1_test	MCC_test
mlp	0.998482	0.998555	0.987131	0.988865	0.999877	0.988207	0.987437
gradient_boosting	0.991725	0.991662	0.892930	0.893851	0.998885	0.929223	0.925619
random_forest	0.941840	0.940680	0.999445	0.992617	0.995074	0.672048	0.686955
decision_tree	0.983297	0.982895	0.856969	0.852215	0.994932	0.859182	0.850110
logistic	0.974901	0.974928	0.686984	0.685548	0.961793	0.770036	0.763492
knn	0.972886	0.965123	0.680645	0.607722	0.948387	0.680906	0.668176
naive_bayes	0.925119	0.925539	0.279126	0.274268	0.811869	0.310858	0.274984
ridge	0.890493	0.891240	0.776853	0.778383	0.000000	0.467081	0.463847

```
plt.figure(figsize = (12,12))
plt.tick_params(axis = 'both', which = 'major', labelsize = 12)
fpr, tpr, _ = metrics.roc_curve(y_test_class, class_models['naive_bayes']['probs'])
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % class_models['naive_bayes']['ROC_AUC_test'])
plt.plot([0, 1], [0, 1], 'k--')  # coin toss line
plt.xlabel('False Positive Rate', fontsize = 14)
plt.ylabel('True Positive Rate', fontsize = 14)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.legend(loc="lower right")
```

```
<matplotlib.legend.Legend at 0x1a34001310>
```



Dimensionality Reduction Methods

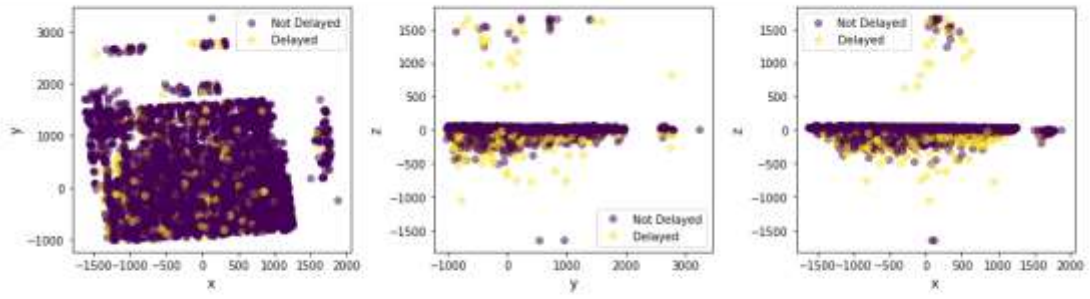
```
for method_name in dimred_methods.keys():
    if method_name != 'vae':
        loudim_data = dimred_methods[method_name]['method'].\
                      fit_transform(X_test_abbrev.values[sample_idx])
    else:
        fitted_model = dimred_methods[method_name]['method'].train(num_steps=1000)
        loudim_data = fitted_model.embed(X_test_abbrev.values[sample_idx])
        dimred_methods[method_name]['fitted'] = fitted_model
    dimred_methods[method_name]['loudim'] = loudim_data
```

```
Starting queues.
WARNING:tensorflow:From /opt/anaconda3/lib/python3.7/site-packages/vae/vae.py:404: start_queue_runners (from tensorflow.python.training.queue_runner_impl) is deprecated and will be removed in a future versio
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
WARNING:tensorflow:`tf.train.start_queue_runners()` was called when no queue runners were defined. You can safely remove the call to this deprecated function.
Reader threads started.
step 0; epoch 0.00 - loss = 7.382, test_loss = 9.677, lr = 0.00100, (0.161 sec/step)
Storing checkpoint to /Users/serg/EXTEND/PROJECTS/InterpretableMLBook/programming/Chapter3/temp ...WARNING:tensorflow:Issue encountered when serializing variables.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'b' has type str, but expected one of: int, long, bool
WARNING:tensorflow:Issue encountered when serializing trainable_variables.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'b' has type str, but expected one of: int, long, bool
 Done.
step 50; epoch 0.00 - loss = 5.285, test_loss = 8.111, lr = 0.00100, (0.123 sec/step)
step 100; epoch 0.01 - loss = 4.019, test_loss = 7.155, lr = 0.00100, (0.140 sec/step)
step 150; epoch 0.01 - loss = 4.166, test_loss = 6.871, lr = 0.00100, (0.142 sec/step)
step 200; epoch 0.02 - loss = 4.209, test_loss = 6.800, lr = 0.00100, (0.148 sec/step)
step 250; epoch 0.02 - loss = 4.642, test_loss = 6.647, lr = 0.00100, (0.148 sec/step)
step 300; epoch 0.03 - loss = 4.157, test_loss = 6.585, lr = 0.00100, (0.147 sec/step)
step 350; epoch 0.03 - loss = 4.173, test_loss = 6.564, lr = 0.00100, (0.145 sec/step)
step 400; epoch 0.04 - loss = 3.935, test_loss = 6.521, lr = 0.00100, (0.152 sec/step)
step 450; epoch 0.04 - loss = 38.228, test_loss = 6.516, lr = 0.00100, (0.148 sec/step)
step 500; epoch 0.05 - loss = 4.016, test_loss = 6.491, lr = 0.00100, (0.153 sec/step)
step 550; epoch 0.05 - loss = 5.147, test_loss = 6.489, lr = 0.00100, (0.148 sec/step)
step 600; epoch 0.06 - loss = 3.814, test_loss = 6.503, lr = 0.00100, (0.159 sec/step)
step 650; epoch 0.06 - loss = 4.514, test_loss = 6.482, lr = 0.00100, (0.145 sec/step)
step 700; epoch 0.07 - loss = 3.978, test_loss = 6.477, lr = 0.00100, (0.136 sec/step)
step 750; epoch 0.07 - loss = 3.722, test_loss = 6.481, lr = 0.00100, (0.134 sec/step)
step 800; epoch 0.07 - loss = 7.441, test_loss = 6.473, lr = 0.00100, (0.138 sec/step)
step 850; epoch 0.08 - loss = 3.749, test_loss = 6.470, lr = 0.00100, (0.136 sec/step)
step 900; epoch 0.08 - loss = 3.879, test_loss = 6.470, lr = 0.00100, (0.138 sec/step)
step 950; epoch 0.09 - loss = 5.024, test_loss = 6.471, lr = 0.00100, (0.136 sec/step)
Reached training step limit of 1000 steps. Stopping.
Storing checkpoint to /Users/serg/EXTEND/PROJECTS/InterpretableMLBook/programming/Chapter3/temp ...WARNING:tensorflow:Issue encountered when serializing variables.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'b' has type str, but expected one of: int, long, bool
WARNING:tensorflow:Issue encountered when serializing trainable_variables.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'b' has type str, but expected one of: int, long, bool
 Done.
```

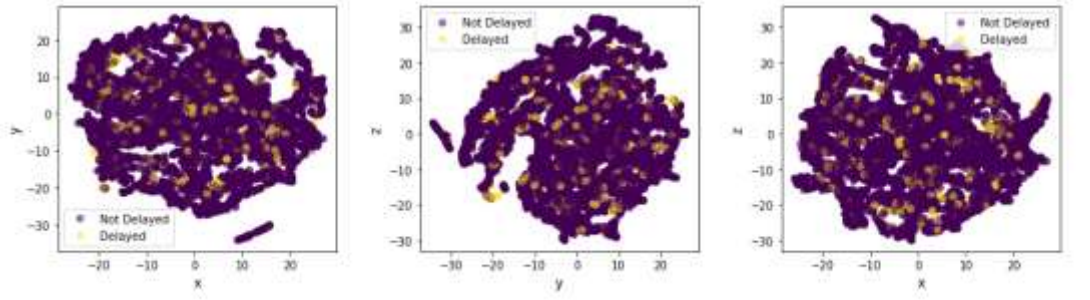
```
y_names = {0:'Not Delayed', 1:'Delayed'}
```

+ Code + Text

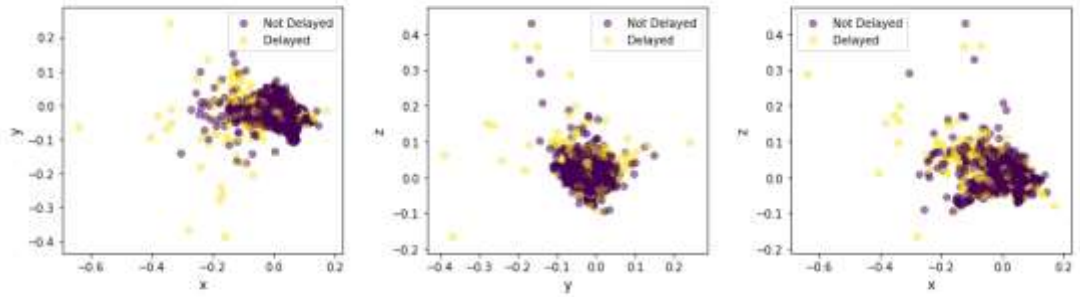
```
mldatasets.plot_3dim_decomposition(dimred_methods['pca']['loudim'],\
                        y_test_class.values[sample_idx], y_names)
```




```
mldatasets.plot_3dim_decomposition(dimred_methods['t-sne']['lowdim'],\
                        y_test_class.values[sample_idx], y_names)
```

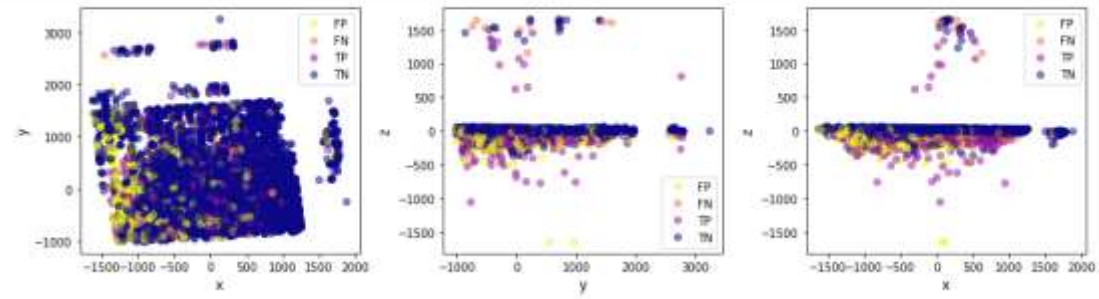


```
mldatasets.plot_3dim_decomposition(dimred_methods['vae']['lowdim'],\
                        y_test_class.values[sample_idx], y_names)
```

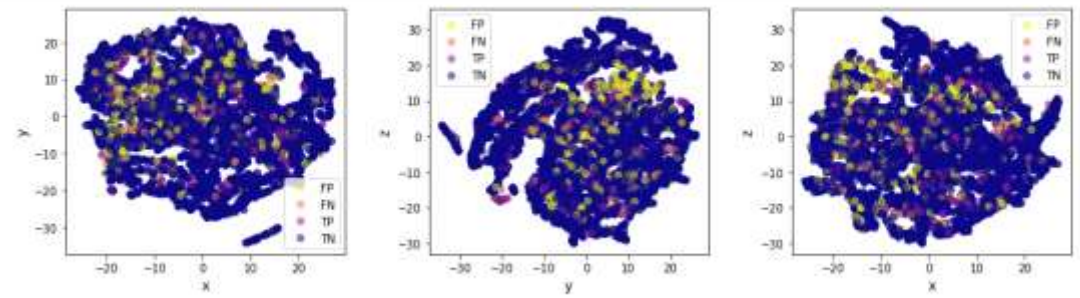



```
y_test_class_samp = y_test_class.values[sample_idx]
y_test_pred_samp = class_models['ridge']['preds'][sample_idx]
error_vector, error_labels = mldatasets.encode_classification_error_vector(y_test_class_samp, y_test_pred_samp)
```

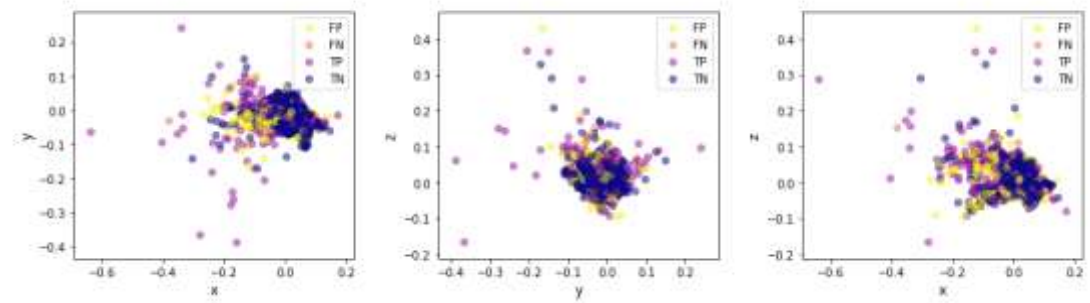
```
mldatasets.plot_3dim_decomposition(dimred_methods['pca']['lowdim'],\
                        error_vector, error_labels)
```




```
mldatasets.plot_3dim_decomposition(dimred_methods['t-sne']['lowdim'],\
                        error_vector, error_labels)
```




```
mldatasets.plot_3dim_decomposition(dimred_methods['vae']['lowdim'],\
                        error_vector, error_labels)
```



Generalised Linear Model

```
coefs_lm = reg_models['linear']['fitted'].coef_
intercept_lm = reg_models['linear']['fitted'].intercept_
print('coefficients:\t%s' % coefs_lm)
print('intercept:\t%s' % intercept_lm)
```

```
coefficients:   [ 4.54984539e-03 -5.25067742e-03  8.94125541e-01 -1.52961053e-02
 -4.69623002e-01  1.25277815e-01 -6.46744472e-04 -1.26240049e-02
  4.50112895e+01  6.76385421e-04 -3.69920254e-04  5.47855860e-04
  3.73866548e-01 -9.06364154e-01 -6.74052666e-01 -9.17411191e-01
 -9.29843952e-01 -3.96621856e-02 -1.79666480e-02 -1.02912927e+00
 -3.94934854e-01]
intercept:      -37.86177932752714
```

```
print('ŷ = %0.2f + %0.4fX₁ + %0.4fX₂ + %0.3fX₃ + ...' %\
      (intercept_lm, coefs_lm[0], coefs_lm[1], coefs_lm[2]))
```

```
ŷ = -37.86 + 0.0045X₁ + -0.0053X₂ + 0.894X₃ + ...
```

```
coef_df = pd.DataFrame({'feature':X_train.columns.values.tolist(),\
                        'coef': coefs_lm})
print(coef_df)
```

	feature	coef
0	CRS_DEP_TIME	0.004550
1	DEP_TIME	-0.005251
2	DEP_DELAY	0.894126
3	DEP_AFPH	-0.015296
4	DEP_RFPH	-0.469623
5	TAXI_OUT	0.125278
6	WHEELS_OFF	-0.000647
7	CRS_ELAPSED_TIME	-0.012624
8	PCT_ELAPSED_TIME	45.011289
9	DISTANCE	0.000676
10	CRS_ARR_TIME	-0.000370
11	ARR_AFPH	0.000548
12	ARR_RFPH	0.373867
13	WEATHER_DELAY	-0.906364
14	NAS_DELAY	-0.674053
15	SECURITY_DELAY	-0.917411
16	LATE_AIRCRAFT_DELAY	-0.929844
17	DEP_MONTH	-0.039662
18	DEP_DOW	-0.017967
19	ORIGIN_HUB	-1.029129

18	DEP_DOW	-0.017967
19	ORIGIN_HUB	-1.029129
20	DEST_HUB	-0.394935

```
linreg_mdl = sm.OLS(y_train_reg, sm.add_constant(X_train))
linreg_mdl = linreg_mdl.fit()
linreg_mdl.summary()
```

OLS Regression Results

Dep. Variable:	CARRIER_DELAY	**R-squared:**	0.921
Model:	OLS	**Adj. R-squared:**	0.921
Method:	Least Squares	**F-statistic:**	4.251e+05
Date:	Wed, 02 Sep 2020	**Prob (F-statistic):**	0.00
Time:	13:32:20	**Log-Likelihood:**	-2.6574e+06
No. Observations:	764597	**AIC:**	5.315e+06
Df Residuals:	764575	**BIC:**	5.315e+06
Df Model:	21		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-37.8618	0.125	-301.763	0.000	-38.108	-37.616
CRS_DEP_TIME	0.0045	7.24e-05	62.872	0.000	0.004	0.005
DEP_TIME	-0.0053	9.19e-05	-57.116	0.000	-0.005	-0.005
DEP_DELAY	0.8941	0.000	2951.056	0.000	0.894	0.895
DEP_AFPH	-0.0153	0.000	-47.725	0.000	-0.016	-0.015
DEP_RFPH	-0.4696	0.017	-27.353	0.000	-0.503	-0.436
TAXI_OUT	0.1253	0.001	104.120	0.000	0.123	0.128
WHEELS_OFF	-0.0006	6.7e-05	-9.646	0.000	-0.001	-0.001
CRS_ELAPSED_TIME	-0.0126	0.001	-19.132	0.000	-0.014	-0.011
PCT_ELAPSED_TIME	45.0113	0.117	384.073	0.000	44.782	45.241
DISTANCE	0.0007	8.02e-05	8.429	0.000	0.001	0.001
CRS_ARR_TIME	-0.0004	2.18e-05	-16.939	0.000	-0.000	-0.000
ARR_AFPH	0.0005	0.000	1.651	0.099	-0.000	0.001
ARR_RFPH	0.3739	0.013	28.386	0.000	0.348	0.400
WEATHER_DELAY	-0.9064	0.001	-995.366	0.000	-0.908	-0.905
NAS_DELAY	-0.6741	0.001	-829.129	0.000	-0.676	-0.672

SECURITY_DELAY	-0.9174	0.005	-167.857	0.000	-0.928	-0.907
LATE_AIRCRAFT_DELAY	-0.9298	0.001	-1827.018	0.000	-0.931	-0.929
DEP_MONTH	-0.0397	0.003	-15.019	0.000	-0.045	-0.034
DEP_DOW	-0.0180	0.004	-4.005	0.000	-0.027	-0.009
ORIGIN_HUB	-1.0291	0.027	-38.589	0.000	-1.081	-0.977
DEST_HUB	-0.3949	0.026	-15.041	0.000	-0.446	-0.343

Omnibus:	211121.387	**Durbin-Watson:**	2.001
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	24359701.834
Skew:	0.098	**Prob(JB):**	0.00
Kurtosis:	30.651	**Cond. No.**	5.69e+04

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.69e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

```
summary_df = linreg_mdl.summary2().tables[1]
summary_df = summary_df.drop(['const']).reset_index().rename(columns={'index':'feature'})
summary_df['t_abs'] = abs(summary_df['t'])
summary_df.sort_values(by='t_abs', ascending=False).style.\
    background_gradient(cmap='plasma_r', low=0, high=0.1, subset=['P>|t|']).\
    background_gradient(cmap='plasma_r', low=0, high=0.1, subset=['t_abs'])
```

	feature	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]	t_abs
2	DEP_DELAY	0.894126	0.000303	2951.055978	0.000000	0.893532	0.894719	2951.055978
16	LATE_AIRCRAFT_DELAY	-0.929844	0.000509	-1827.018082	0.000000	-0.930841	-0.928846	1827.018082
13	WEATHER_DELAY	-0.906364	0.000911	-995.366423	0.000000	-0.908149	-0.904579	995.366423
14	NAS_DELAY	-0.674053	0.000813	-829.128657	0.000000	-0.675646	-0.672459	829.128657
8	PCT_ELAPSED_TIME	45.011289	0.117195	384.072566	0.000000	44.781592	45.240987	384.072566
15	SECURITY_DELAY	-0.917411	0.005465	-167.857085	0.000000	-0.928123	-0.906699	167.857085
5	TAXI_OUT	0.125278	0.001203	104.119579	0.000000	0.122920	0.127636	104.119579
0	CRS_DEP_TIME	0.004550	0.000072	62.871693	0.000000	0.004408	0.004692	62.871693
1	DEP_TIME	-0.005251	0.000092	-57.115895	0.000000	-0.005431	-0.005070	57.115895
3	DEP_AFPH	-0.015296	0.000321	-47.724506	0.000000	-0.015924	-0.014668	47.724506
19	ORIGIN_HUB	-1.029129	0.026669	-38.589411	0.000000	-1.081399	-0.976860	38.589411
12	ARR_RFPH	0.373867	0.013171	28.386031	0.000000	0.348052	0.399681	28.386031
4	DEP_RFPH	-0.469623	0.017169	-27.353179	0.000000	-0.503273	-0.435973	27.353179
7	CRS_ELAPSED_TIME	-0.012624	0.000660	-19.131516	0.000000	-0.013917	-0.011331	19.131516
10	CRS_ARR_TIME	-0.000370	0.000022	-16.938661	0.000000	-0.000413	-0.000327	16.938661
20	DEST_HUB	-0.394935	0.026256	-15.041459	0.000000	-0.446397	-0.343473	15.041459
17	DEP_MONTH	-0.039662	0.002641	-15.018808	0.000000	-0.044838	-0.034486	15.018808
6	WHEELS_OFF	-0.000647	0.000067	-9.646104	0.000000	-0.000778	-0.000515	9.646104
9	DISTANCE	0.000676	0.000080	8.428835	0.000000	0.000519	0.000834	8.428835
18	DEP_DOW	-0.017967	0.004487	-4.004561	0.000062	-0.026760	-0.009173	4.004561
11	ARR_AFPH	0.000548	0.000332	1.650788	0.098782	-0.000103	0.001198	1.650788

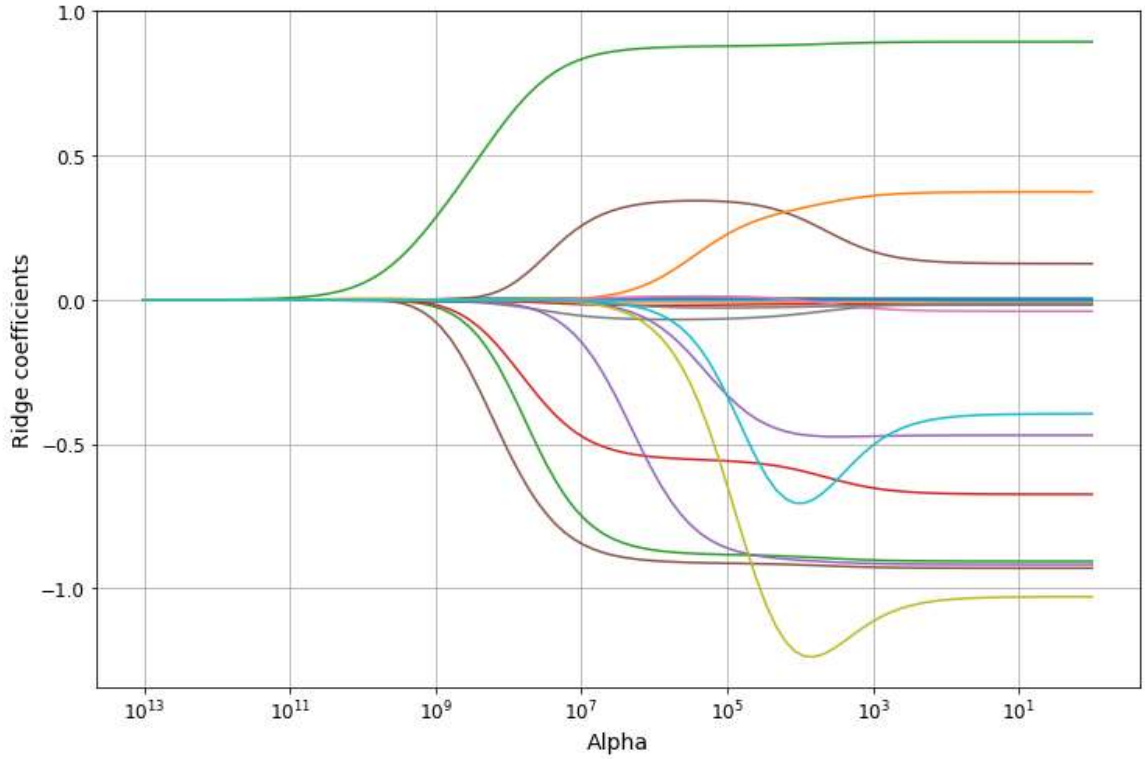
Ridge Regression:

```
coefs_ridge = reg_models['ridge']['fitted'].coef_
coef_ridge_df = pd.DataFrame({'feature':X_train.columns.values.tolist(),\
                          'coef_linear': coefs_lm,\
                          'coef_ridge': coefs_ridge})
coef_ridge_df.style.\
    background_gradient(cmap='viridis_r', low=0.3, high=0.2, axis=1)
```

	feature	coef_linear	coef_ridge
0	CRS_DEP_TIME	0.004550	0.005373
1	DEP_TIME	-0.005251	-0.003627
2	DEP_DELAY	0.894126	0.894013
3	DEP_AFPH	-0.015296	-0.015322
4	DEP_RFPH	-0.469623	-0.469625
5	TAXI_OUT	0.125278	0.125244
6	WHEELS_OFF	-0.000647	0.001053
7	CRS_ELAPSED_TIME	-0.012624	-0.012727
8	PCT_ELAPSED_TIME	45.011289	45.011279
9	DISTANCE	0.000676	0.000165
10	CRS_ARR_TIME	-0.000370	-0.000148
11	ARR_AFPH	0.000548	0.000543
12	ARR_RFPH	0.373867	0.373868
13	WEATHER_DELAY	-0.906364	-0.906506
14	NAS_DELAY	-0.674053	-0.673995
15	SECURITY_DELAY	-0.917411	-0.917405
16	LATE_AIRCRAFT_DELAY	-0.929844	-0.929629
17	DEP_MONTH	-0.039662	-0.039662
18	DEP_DOW	-0.017967	-0.017968
19	ORIGIN_HUB	-1.029129	-1.029130
20	DEST_HUB	-0.394935	-0.394935

```
num_alphas = 100
alphas = np.logspace(0, 13, num_alphas)
alphas_coefs = []
for alpha in alphas:
    ridge = linear_model.Ridge(alpha=alpha).fit(X_train, y_train_reg)
    alphas_coefs.append(np.concatenate((ridge.coef_[:8],ridge.coef_[9:])))
```

```
plt.figure(figsize = (12,8))
plt.gca().invert_xaxis()
plt.tick_params(axis = 'both', which = 'major', labelsize = 12)
plt.plot(alphas, alphas_coefs)
plt.xscale("log")
plt.xlabel('Alpha', fontsize = 14)
plt.ylabel('Ridge coefficients', fontsize = 14)
plt.grid()
plt.show()
```



Polynomial Regression:

```
print(reg_models['linear_poly']['fitted'].get_params()['linearregression'].coef_.shape[0])
```

```
253
```

```
print(reg_models['linear_interact']['fitted'].get_params()['linearregression'].coef_.shape[0])
```

```
232
```

Logistic Regression:

```
coefs_log = class_models['logistic']['fitted'].coef_
intercept_log = class_models['logistic']['fitted'].intercept_
print('coefficients:\t%s' % coefs_log)
print('intercept:\t%s' % intercept_log)
```

```
coefficients:   [[-1.23151812e-03  2.93774704e-04  1.61338034e-01  1.77493349e-03
  -2.15915422e-03 -4.35450242e-03  1.27984682e-04 -5.17720785e-02
  -1.76310069e-03  5.61580071e-03 -2.53698323e-04 -6.79027014e-03
  -2.74571712e-03 -1.53557236e-01 -1.21227428e-01 -5.99414743e-03
  -1.65188544e-01 -1.31214181e-02 -5.33132240e-03  4.78845293e-04
  -2.07097380e-03]]
intercept:      [-0.00230901]
```

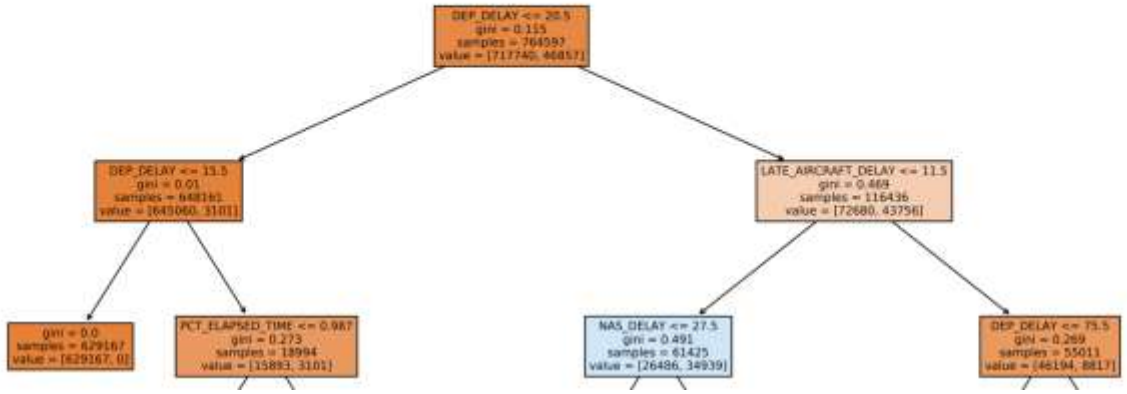
```
stdv = np.std(X_train, 0)
abs(coefs_log.reshape(21,) * stdv).sort_values(ascending=False)
```

```
DEP_DELAY              7.141532
CRS_ELAPSED_TIME       4.109450
LATE_AIRCRAFT_DELAY    4.082085
DISTANCE               3.647053
NAS_DELAY              1.670519
WEATHER_DELAY          1.604633
CRS_DEP_TIME           0.617492
ARR_AFPH               0.240126
DEP_TIME               0.151267
CRS_ARR_TIME           0.133986
WHEELS_OFF             0.066116
DEP_AFPH               0.062930
DEP_MONTH              0.044779
TAXI_OUT               0.042975
DEP_DOW                0.010654
SECURITY_DELAY         0.009825
ARR_RFPH               0.001981
DEP_RFPH               0.001220
DEST_HUB               0.001007
ORIGIN_HUB             0.000233
PCT_ELAPSED_TIME       0.000186
dtype: float64
```

Decision Tree:

```
fig, axes = plt.subplots(nrows = 1, ncols = 1, figsize = (16,8), dpi=600)
tree.plot_tree(class_models['decision_tree'][0].fitted,\
               feature_names=X_train.columns.values.tolist(), filled = True, max_depth=2)
fig.show()
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4: UserWarning: Matplotlib is currently using module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot show the figure.
  after removing the cwd from sys.path.
```



```
text_tree = tree.export_text(class_models['decision_tree']['fitted'],\
               feature_names=X_train.columns.values.tolist())
print(text_tree)
```

```
|--- DEP_DELAY <= 20.50
|   |--- DEP_DELAY <= 15.50
|   |   |--- class: 0
|   |--- DEP_DELAY >  15.50
|   |   |--- PCT_ELAPSED_TIME <= 0.99
|   |   |   |--- PCT_ELAPSED_TIME <= 0.98
|   |   |   |   |--- PCT_ELAPSED_TIME <= 0.96
|   |   |   |   |   |--- CRS_ELAPSED_TIME <= 65.50
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME <= 0.94
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME >  0.94
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- CRS_ELAPSED_TIME >  65.50
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME <= 0.95
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME >  0.95
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- PCT_ELAPSED_TIME >  0.96
|   |   |   |   |   |--- CRS_ELAPSED_TIME <= 140.50
|   |   |   |   |   |   |--- DEP_DELAY <= 18.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- DEP_DELAY >  18.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- CRS_ELAPSED_TIME >  140.50
|   |   |   |   |   |   |--- DEP_DELAY <= 19.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- DEP_DELAY >  19.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |--- PCT_ELAPSED_TIME >  0.98
|   |   |   |   |--- DEP_DELAY <= 18.50
|   |   |   |   |   |--- DISTANCE <= 326.50
|   |   |   |   |   |   |--- LATE_AIRCRAFT_DELAY <= 0.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- LATE_AIRCRAFT_DELAY >  0.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- DISTANCE >  326.50
|   |   |   |   |   |   |--- DEP_DELAY <= 17.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- DEP_DELAY >  17.50
```

```
|   |   |   |   |   |   |--- DEP_DELAY >  17.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- DEP_DELAY >  18.50
|   |   |   |   |   |   |--- LATE_AIRCRAFT_DELAY <= 1.50
|   |   |   |   |   |   |   |--- DISTANCE <= 1358.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- DISTANCE >  1358.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- LATE_AIRCRAFT_DELAY >  1.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |--- PCT_ELAPSED_TIME >  0.99
|   |   |   |   |--- LATE_AIRCRAFT_DELAY <= 1.50
|   |   |   |   |   |--- WEATHER_DELAY <= 2.00
|   |   |   |   |   |   |--- NAS_DELAY <= 17.50
|   |   |   |   |   |   |   |--- PCT_ELAPSED_TIME <= 1.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- PCT_ELAPSED_TIME >  1.00
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- NAS_DELAY >  17.50
|   |   |   |   |   |   |   |--- PCT_ELAPSED_TIME <= 1.09
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- PCT_ELAPSED_TIME >  1.09
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- WEATHER_DELAY >  2.00
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- LATE_AIRCRAFT_DELAY >  1.50
|   |   |   |   |   |--- LATE_AIRCRAFT_DELAY <= 3.50
|   |   |   |   |   |   |--- DEP_DELAY <= 18.50
|   |   |   |   |   |   |   |--- DISTANCE <= 153.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- DISTANCE >  153.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- DEP_DELAY >  18.50
|   |   |   |   |   |   |   |--- WEATHER_DELAY <= 2.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- WEATHER_DELAY >  2.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- LATE_AIRCRAFT_DELAY >  3.50
|   |   |   |   |   |   |--- LATE_AIRCRAFT_DELAY <= 4.50
|   |   |   |   |   |   |   |--- DEP_DELAY <= 19.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- DEP_DELAY >  19.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- LATE_AIRCRAFT_DELAY >  4.50
```

```
|   |   |   |   |   |   |--- class: 0
|--- DEP_DELAY >  20.50
|   |--- LATE_AIRCRAFT_DELAY <= 11.50
|   |   |--- NAS_DELAY <= 27.50
|   |   |   |--- DEP_DELAY <= 35.50
|   |   |   |   |--- PCT_ELAPSED_TIME <= 0.96
|   |   |   |   |   |--- DEP_DELAY <= 28.50
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME <= 0.93
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME >  0.93
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- DEP_DELAY >  28.50
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME <= 0.92
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME >  0.92
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- PCT_ELAPSED_TIME >  0.96
|   |   |   |   |   |--- WEATHER_DELAY <= 4.50
|   |   |   |   |   |   |--- LATE_AIRCRAFT_DELAY <= 6.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- LATE_AIRCRAFT_DELAY >  6.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- WEATHER_DELAY >  4.50
|   |   |   |   |   |   |--- WEATHER_DELAY <= 10.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- WEATHER_DELAY >  10.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |--- DEP_DELAY >  35.50
|   |   |   |   |--- WEATHER_DELAY <= 16.50
|   |   |   |   |   |--- DEP_DELAY <= 44.50
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME <= 0.93
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME >  0.93
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- DEP_DELAY >  44.50
|   |   |   |   |   |   |--- SECURITY_DELAY <= 20.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- SECURITY_DELAY >  20.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- WEATHER_DELAY >  16.50
|   |   |   |   |   |--- WEATHER_DELAY <= 23.50
|   |   |   |   |   |   |--- DEP_DELAY <= 57.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- DEP_DELAY >  57.50
```

```
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- WEATHER_DELAY >  23.50
|   |   |   |   |   |   |   |--- DEP_DELAY <= 88.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- DEP_DELAY >  88.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |--- NAS_DELAY >  27.50
|   |   |   |   |--- PCT_ELAPSED_TIME <= 1.11
|   |   |   |   |   |--- NAS_DELAY <= 31.50
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME <= 1.07
|   |   |   |   |   |   |   |--- DEP_DELAY <= 69.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- DEP_DELAY >  69.00
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME >  1.07
|   |   |   |   |   |   |   |--- WEATHER_DELAY <= 10.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- WEATHER_DELAY >  10.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- NAS_DELAY >  31.50
|   |   |   |   |   |   |--- DEP_DELAY <= 471.50
|   |   |   |   |   |   |   |--- CRS_ELAPSED_TIME <= 420.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- CRS_ELAPSED_TIME >  420.00
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- DEP_DELAY >  471.50
|   |   |   |   |   |   |   |--- NAS_DELAY <= 388.00
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- NAS_DELAY >  388.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- PCT_ELAPSED_TIME >  1.11
|   |   |   |   |   |--- NAS_DELAY <= 64.50
|   |   |   |   |   |   |--- WEATHER_DELAY <= 20.50
|   |   |   |   |   |   |   |--- DEP_DELAY <= 43.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- DEP_DELAY >  43.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- WEATHER_DELAY >  20.50
|   |   |   |   |   |   |   |--- WHEELS_OFF <= 36.00
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- WHEELS_OFF >  36.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- NAS_DELAY >  64.50
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME <= 1.44
```

```
|   |   |   |   |   |   |--- NAS_DELAY <= 78.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- NAS_DELAY >  78.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- PCT_ELAPSED_TIME >  1.44
|   |   |   |   |   |   |--- NAS_DELAY <= 119.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- NAS_DELAY >  119.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |--- LATE_AIRCRAFT_DELAY >  11.50
|   |   |   |--- DEP_DELAY <= 75.50
|   |   |   |   |--- DEP_DELAY <= 41.50
|   |   |   |   |   |--- LATE_AIRCRAFT_DELAY <= 14.50
|   |   |   |   |   |   |--- DEP_DELAY <= 29.50
|   |   |   |   |   |   |   |--- DEP_DELAY <= 27.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- DEP_DELAY >  27.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- DEP_DELAY >  29.50
|   |   |   |   |   |   |   |--- PCT_ELAPSED_TIME <= 0.97
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- PCT_ELAPSED_TIME >  0.97
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- LATE_AIRCRAFT_DELAY >  14.50
|   |   |   |   |   |   |--- LATE_AIRCRAFT_DELAY <= 20.50
|   |   |   |   |   |   |   |--- DEP_DELAY <= 32.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- DEP_DELAY >  32.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- LATE_AIRCRAFT_DELAY >  20.50
|   |   |   |   |   |   |   |--- DEP_DELAY <= 38.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- DEP_DELAY >  38.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- DEP_DELAY >  41.50
|   |   |   |   |   |--- LATE_AIRCRAFT_DELAY <= 29.50
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME <= 0.94
|   |   |   |   |   |   |   |--- DEP_DELAY <= 55.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- DEP_DELAY >  55.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME >  0.94
|   |   |   |   |   |   |   |--- WEATHER_DELAY <= 0.50
|   |   |   |   |   |   |   |   |--- class: 1
```

```
|   |   |   |   |--- DEP_DELAY >  41.50
|   |   |   |   |   |--- LATE_AIRCRAFT_DELAY <= 29.50
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME <= 0.94
|   |   |   |   |   |   |   |--- DEP_DELAY <= 55.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- DEP_DELAY >  55.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- PCT_ELAPSED_TIME >  0.94
|   |   |   |   |   |   |   |--- WEATHER_DELAY <= 0.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- WEATHER_DELAY >  0.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- LATE_AIRCRAFT_DELAY >  29.50
|   |   |   |   |   |   |--- LATE_AIRCRAFT_DELAY <= 38.50
|   |   |   |   |   |   |   |--- DEP_DELAY <= 59.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- DEP_DELAY >  59.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- LATE_AIRCRAFT_DELAY >  38.50
|   |   |   |   |   |   |   |--- DEP_DELAY <= 60.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- DEP_DELAY >  60.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |--- DEP_DELAY >  75.50
|   |   |   |   |--- LATE_AIRCRAFT_DELAY <= 60.50
|   |   |   |   |   |--- WEATHER_DELAY <= 0.50
|   |   |   |   |   |   |--- NAS_DELAY <= 38.50
|   |   |   |   |   |   |   |--- DEP_DELAY <= 88.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- DEP_DELAY >  88.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- NAS_DELAY >  38.50
|   |   |   |   |   |   |   |--- TAXI_OUT <= 63.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- TAXI_OUT >  63.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- WEATHER_DELAY >  0.50
|   |   |   |   |   |   |--- WEATHER_DELAY <= 18.50
|   |   |   |   |   |   |   |--- LATE_AIRCRAFT_DELAY <= 31.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- LATE_AIRCRAFT_DELAY >  31.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- WEATHER_DELAY >  18.50
|   |   |   |   |   |   |   |--- DEP_AFPH <= 99.64
```

```
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- DEP_AFPH >  99.64
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |--- LATE_AIRCRAFT_DELAY >  60.50
|   |   |   |   |--- DEP_DELAY <= 114.50
|   |   |   |   |   |--- LATE_AIRCRAFT_DELAY <= 71.50
|   |   |   |   |   |   |--- DEP_DELAY <= 95.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- DEP_DELAY >  95.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- LATE_AIRCRAFT_DELAY >  71.50
|   |   |   |   |   |   |--- DEP_DELAY <= 96.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- DEP_DELAY >  96.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- DEP_DELAY >  114.50
|   |   |   |   |   |--- LATE_AIRCRAFT_DELAY <= 98.50
|   |   |   |   |   |   |--- WEATHER_DELAY <= 1.00
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- WEATHER_DELAY >  1.00
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- LATE_AIRCRAFT_DELAY >  98.50
|   |   |   |   |   |   |--- DEP_DELAY <= 171.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- DEP_DELAY >  171.50
|   |   |   |   |   |   |   |--- class: 0
```

```
dt_imp_df = pd.DataFrame({'feature':X_train.columns.values.tolist(),\
                          'importance': class_models['decision_tree']['fitted'].feature_importances_}).\
            sort_values(by='importance', ascending=False)
dt_imp_df
```

	feature	importance
2	DEP_DELAY	0.527482
16	LATE_AIRCRAFT_DELAY	0.199153
8	PCT_ELAPSED_TIME	0.105381
13	WEATHER_DELAY	0.101649
14	NAS_DELAY	0.062732
15	SECURITY_DELAY	0.001998
9	DISTANCE	0.001019
7	CRS_ELAPSED_TIME	0.000281
5	TAXI_OUT	0.000239
6	WHEELS_OFF	0.000035
3	DEP_AFPH	0.000031
0	CRS_DEP_TIME	0.000000
19	ORIGIN_HUB	0.000000
18	DEP_DOW	0.000000
17	DEP_MONTH	0.000000
10	CRS_ARR_TIME	0.000000
12	ARR_RFPH	0.000000
11	ARR_AFPH	0.000000
1	DEP_TIME	0.000000

11	ARR_AFPH	0.000000
1	DEP_TIME	0.000000
4	DEP_RFPH	0.000000
20	DEST_HUB	0.000000

leFit

```
rulefit_df = reg_models['rulefit']['fitted'].get_rules()
rulefit_df = rulefit_df[rulefit_df.coef != 0].sort_values(by="importance", ascending=False)
rulefit_df
```

	rule	type	coef	support	importance
56	DEP_DELAY > 344.0 & WEATHER_DELAY <= 166.0 & L...	rule	222.054419	0.001684	9.103331
23	LATE_AIRCRAFT_DELAY <= 333.5 & DEP_DELAY > 477.5	rule	172.380029	0.001122	5.771707
16	LATE_AIRCRAFT_DELAY	linear	-0.386460	1.000000	4.528198
2	DEP_DELAY	linear	0.163546	1.000000	4.278776
67	DEP_DELAY > 1206.0	rule	290.051584	0.000187	3.966616
...	...	...	...	...	...
62	DEP_DELAY <= 477.5 & DEP_DELAY > 218.5	rule	0.304649	0.006173	0.023861
115	DEP_DELAY <= 490.5 & DEP_DELAY > 58.5 & LATE_A...	rule	0.120538	0.032735	0.021449
18	DEP_DOW	linear	0.009917	1.000000	0.019817
163	DEP_DELAY <= 136.5 & DEP_DELAY <= 48.5	rule	-0.046215	0.922933	0.012325
72	DEP_DELAY <= 48.5 & DEP_DELAY <= 534.5	rule	-0.007695	0.921811	0.002066

82 rows × 5 columns

Nearest Neighbors:

KNN

```
print(X_test.loc[721043,:])
```

```
CRS_DEP_TIME            655.000000
DEP_TIME               1055.000000
DEP_DELAY               240.000000
DEP_AFPH                 90.800000
DEP_RFPH                  0.890196
TAXI_OUT                 35.000000
WHEELS_OFF             1130.000000
CRS_ELAPSED_TIME        259.000000
PCT_ELAPSED_TIME          1.084942
DISTANCE               1660.000000
CRS_ARR_TIME            914.000000
ARR_AFPH                 40.434783
ARR_RFPH                  1.064073
WEATHER_DELAY             0.000000
NAS_DELAY                22.000000
SECURITY_DELAY            0.000000
LATE_AIRCRAFT_DELAY     221.000000
DEP_MONTH                10.000000
DEP_DOW                   4.000000
ORIGIN_HUB                1.000000
DEST_HUB                  0.000000
Name: 721043, dtype: float64
```

```
print(y_test_class[721043])
```

```
1
```

```
print(y_test_class[721043])
```

```
1
```

```
print(class_models['knn']['preds'][X_test.index.get_loc(721043)])
```

```
0
```

```
print(class_models['knn']['fitted'].kneighbors(X_test.loc[721043,:].values.reshape(1,21), 7))
```

```
(array([[143.3160128 , 173.90740076, 192.66705727, 211.57109221,
        243.57211853, 259.61593993, 259.77507391]]), array([[105172, 571912,  73409,  89450,  77474, 705972, 706911]]))
```

```
print(y_train_class.iloc[[105172, 571912,  73409,  89450,  77474, 705972, 706911]])
```

```
3813      0
229062    1
283316    0
385831    0
581905    1
726784    1
179364    0
Name: CARRIER_DELAY, dtype: int64
```

+ Code + Text

```
print(class_models['knn']['fitted'].effective_metric_)
```

```
euclidean
```

Naive Bayes:
Gaussian Naive Bayes

```
print(class_models['naive_bayes']['fitted'].class_prior_)
```

```
[0.93871674 0.06128326]
```

```
print(class_models['naive_bayes']['fitted'].sigma_)
```

```
[[2.50123026e+05 2.61324730e+05 9.21572605e+02 1.26123968e+03
  2.08339528e-01 9.58074414e+01 2.62606651e+05 6.30102550e+03
  1.13475535e-02 4.22470414e+05 2.75433641e+05 1.25314386e+03
  3.48655340e-01 1.11234714e+02 1.91877186e+02 2.80302201e+00
  5.06561612e+02 1.17346654e+01 3.99122491e+00 2.39015406e-01
  2.34996222e-01]
 [2.60629652e+05 2.96009867e+05 1.19307931e+04 1.14839167e+03
  1.99929921e+00 1.20404927e+02 3.08568277e+05 6.29066219e+03
  1.38936741e-02 4.10198938e+05 3.28574000e+05 1.09023147e+03
  3.08997044e+00 7.79140423e+01 1.56184090e+02 9.12112286e-01
  2.11279954e+03 1.02712368e+01 4.02943162e+00 1.77750796e-01
  2.50208354e-01]]
```

```
print(class_models['naive_bayes']['fitted'].theta_)
```

```
[[1.30740577e+03 1.31006271e+03 5.14196506e+00 5.45864877e+01
  1.09377996e+00 1.87120810e+01 1.33552258e+03 1.70734929e+02
  9.71131781e-01 1.01824369e+03 1.48438931e+03 5.39873058e+01
  1.09644787e+00 7.39971299e-01 2.85434558e+00 2.41814585e-02
  4.14674395e+00 6.55045281e+00 2.95035528e+00 6.06800513e-01
  6.24199571e-01]
 [1.41305545e+03 1.48087887e+03 8.45867640e+01 6.14731036e+01
  1.25429654e+00 1.99378321e+01 1.49409412e+03 1.72229998e+02
  9.83974416e-01 1.04363666e+03 1.54821862e+03 4.26486417e+01
  1.36373798e+00 4.50733082e-01 4.71991378e+00 2.11281132e-02
  1.40744819e+01 6.73367907e+00 3.04251232e+00 7.69575517e-01
  4.85391724e-01]]
```

Explainable Boosting Machine (EBM)

```
#Make new abbreviated versions of datasets
feature_samp = ['DEP_DELAY', 'LATE_AIRCRAFT_DELAY', 'PCT_ELAPSED_TIME', 'WEATHER_DELAY',\
                'NAS_DELAY', 'SECURITY_DELAY', 'DISTANCE', 'CRS_ELAPSED_TIME']
X_train_abbrev2 = X_train[feature_samp]
X_test_abbrev2 = X_test[feature_samp]
#For sampling among observations
np.random.seed(rand)
sample2_size = 0.1
sample2_idx = np.random.choice(X_train.shape[0], math.ceil(X_train.shape[0]*sample2_size), replace=False)
```

```
ebm_mdl = ExplainableBoostingClassifier()
ebm_mdl.fit(X_train_abbrev2.iloc[sample2_idx], y_train_class.iloc[sample2_idx])
```

```
ExplainableBoostingClassifier(binning='quantile', early_stopping_rounds=50,
                              early_stopping_tolerance=0,
                              feature_names=['DEP_DELAY', 'LATE_AIRCRAFT_DELAY',
                                             'PCT_ELAPSED_TIME',
                                             'WEATHER_DELAY', 'NAS_DELAY',
                                             'SECURITY_DELAY', 'DISTANCE',
                                             'CRS_ELAPSED_TIME'],
                              feature_types=['continuous', 'continuous',
                                             'continuous', 'continuous',
                                             'continuous', 'continuous',
                                             'continuous', 'continuous'],
                              inner_bags=0, interactions=0, learning_rate=0.01,
                              mains='all', max_bins=255, max_leaves=3,
                              max_rounds=5000, min_samples_leaf=2, n_jobs=-2,
                              outer_bags=16, random_state=42,
                              validation_size=0.15)
```

Skopped Rules:

```
sr_mdl = SkopeRules(n_estimators=200, precision_min=0.2, recall_min=0.01,\
                    n_jobs=-1, random_state=rand, max_depth=7,\
                    feature_names=X_train_abbrev2.columns)
sr_mdl.fit(X_train_abbrev2.iloc[sample2_idx], y_train_class.iloc[sample2_idx])
```

```
SkopeRules(bootstrap=False, bootstrap_features=False,
          feature_names=Index(['DEP_DELAY', 'LATE_AIRCRAFT_DELAY', 'PCT_ELAPSED_TIME', 'WEATHER_DELAY',
       'NAS_DELAY', 'SECURITY_DELAY', 'DISTANCE', 'CRS_ELAPSED_TIME'],
      dtype='object'),
          max_depth=7, max_depth_duplication=None, max_features=1.0,
          max_samples=0.8, max_samples_features=1.0, min_samples_split=2,
          n_estimators=200, n_jobs=-1, precision_min=0.2, random_state=9,
          recall_min=0.01, verbose=0)
```

```
sr_y_test_prob = sr_mdl.score_top_rules(X_test_abbrev2.iloc[sample_idx])
sr_y_test_pred = np.where(sr_y_test_prob > 0.5, 1, 0)
```

```
print(len(sr_mdl.rules_))
```

```
1517
```

```
print(sr_mdl.rules_[0:5])
```

```
[('DEP_DELAY > 39.5 and LATE_AIRCRAFT_DELAY <= 12.5 and WEATHER_DELAY <= 12.0 and NAS_DELAY <= 27.5 and SECURITY_DELAY <= 8.5', (0.95
```

```
print('actual: %s, predicted: %s' % (y_test_class.iloc[76], sr_y_test_pred[76]))
```

```
actual: 1, predicted: 0
```

```
print(sr_mdl.decision_function(X_test_abbrev2.iloc[76:77]))
```

```
[18.23328729]
```

```
print('accuracy: %.3g, recall: %.3g, roc auc: %.3g, f1: %.3g, mcc: %.3g' %\
      (metrics.accuracy_score(y_test_class.iloc[sample_idx], sr_y_test_pred),\
       metrics.recall_score(y_test_class.iloc[sample_idx], sr_y_test_pred),\
       metrics.roc_auc_score(y_test_class.iloc[sample_idx], sr_y_test_prob),\
       metrics.f1_score(y_test_class.iloc[sample_idx], sr_y_test_pred),\
       metrics.matthews_corrcoef(y_test_class.iloc[sample_idx], sr_y_test_pred)))
```

```
accuracy: 0.961, recall: 0.981, roc auc: 0.984, f1: 0.748, mcc: 0.753
```

Chapter 4 (Birth Order):

```
!pip install --upgrade pandas numpy scikit-learn matplotlib
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages (1.3.5)
Collecting pandas
  Downloading pandas-1.5.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.2/12.2 MB 48.4 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (1.21.6)
Collecting numpy
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 33.1 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.8/dist-packages (1.0.2)
Collecting scikit-learn
  Downloading scikit_learn-1.2.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (9.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.8/9.8 MB 42.3 MB/s eta 0:00:00
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (3.2.2)
Collecting matplotlib
  Downloading matplotlib-3.6.3-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (9.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.4/9.4 MB 13.5 MB/s eta 0:00:00
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (3.1.0)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.7.3)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (1.4.4)
Collecting contourpy>=1.0.1
  Downloading contourpy-1.0.7-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (300 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 300.0/300.0 KB 12.3 MB/s eta 0:00:00
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (7.1.2)
Collecting fonttools>=4.22.0
  Downloading fonttools-4.38.0-py3-none-any.whl (965 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 965.4/965.4 KB 16.6 MB/s eta 0:00:00
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (23.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (0.11.0)
Requirement already satisfied: pyparsing>=2.2.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (3.0.9)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.8.1->pandas) (1.15.0)
Collecting numpy
  Downloading numpy-1.22.4-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.9/16.9 MB 48.3 MB/s eta 0:00:00
Installing collected packages: numpy, fonttools, pandas, contourpy, scikit-learn, matplotlib
  Attempting uninstall: numpy
    Found existing installation: numpy 1.21.6
```

```
!pip install --upgrade machine-learning-datasets
!pip install git+https://github.com/SauceCat/PDPbox.git
!pip install --upgrade pycebox
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting machine-learning-datasets
  Downloading machine_learning_datasets-0.1.16.6-py3-none-any.whl (25 kB)
Requirement already satisfied: tqdm<5.0.0,>=4.41.1 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (4.64.1)
Requirement already satisfied: opencv-python<5.0.0,>=4.5.1 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (4.6.0.66)
Collecting alibi<0.6.0,>=0.5.5
  Downloading alibi-0.5.8-py3-none-any.whl (312 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 312.5/312.5 KB 8.4 MB/s eta 0:00:00
Requirement already satisfied: seaborn<0.12.0,>=0.11.1 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (0.11.2)
Collecting aif360<0.4.0,>=0.3.0
  Downloading aif360-0.3.0-py3-none-any.whl (165 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 165.6/165.6 KB 16.7 MB/s eta 0:00:00
Collecting pycebox<0.0.2,>=0.0.1
  Downloading pycebox-0.0.1.tar.gz (4.5 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: mlxtend<0.15.0,>=0.14.0 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (0.14.0)
Requirement already satisfied: matplotlib<4.0.0,>=3.2.2 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (3.6.3)
Requirement already satisfied: pandas<2.0.0,>=1.1.5 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (1.5.3)
Collecting scikit-learn<0.23.0,>=0.22.2.post1
  Downloading scikit_learn-0.22.2.post1-cp38-cp38-manylinux1_x86_64.whl (7.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.0/7.0 MB 24.0 MB/s eta 0:00:00
Collecting statsmodels<0.11.0,>=0.10.2
  Downloading statsmodels-0.10.2-cp38-cp38-manylinux1_x86_64.whl (8.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.1/8.1 MB 50.2 MB/s eta 0:00:00
Requirement already satisfied: numpy<2.0.0,>=1.19.5 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (1.22.4)
Collecting pathlib2<3.0.0,>=2.3.5
  Downloading pathlib2-2.3.7.post1-py2.py3-none-any.whl (18 kB)
Requirement already satisfied: scipy<2.0.0,>=1.4.1 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (1.7.3)
Requirement already satisfied: Pillow<9.0,>=5.4.1 in /usr/local/lib/python3.8/dist-packages (from alibi<0.6.0,>=0.5.5->machine-learning-datasets) (7.1.2)
Collecting attrs<21.0.0,>=19.3.0
  Downloading attrs-20.3.0-py2.py3-none-any.whl (49 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.3/49.3 KB 5.7 MB/s eta 0:00:00
Requirement already satisfied: dill<0.4.0,>=0.3.0 in /usr/local/lib/python3.8/dist-packages (from alibi<0.6.0,>=0.5.5->machine-learning-datasets) (0.3.6)
Requirement already satisfied: spacy[lookups]<4.0.0,>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from alibi<0.6.0,>=0.5.5->machine-learning-datasets) (3.4
Collecting tensorflow<2.5.0,>=2.0.0
```

```
pip install pdpbox==0.2.0
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pdpbox==0.2.0
  Downloading PDPbox-0.2.0.tar.gz (57.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.7/57.7 MB 5.9 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages (from pdpbox==0.2.0) (1.5.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from pdpbox==0.2.0) (1.22.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-packages (from pdpbox==0.2.0) (1.4.1)
Requirement already satisfied: matplotlib>=2.1.2 in /usr/local/lib/python3.8/dist-packages (from pdpbox==0.2.0) (3.6.3)
Requirement already satisfied: joblib in /usr/local/lib/python3.8/dist-packages (from pdpbox==0.2.0) (1.2.0)
Requirement already satisfied: psutil in /usr/local/lib/python3.8/dist-packages (from pdpbox==0.2.0) (5.4.8)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.8/dist-packages (from pdpbox==0.2.0) (0.22.2.post1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.8/dist-packages (from matplotlib>=2.1.2->pdpbox==0.2.0) (4.38.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib>=2.1.2->pdpbox==0.2.0) (0.11.0)
Requirement already satisfied: pyparsing>=2.2.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib>=2.1.2->pdpbox==0.2.0) (3.0.9)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib>=2.1.2->pdpbox==0.2.0) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.8/dist-packages (from matplotlib>=2.1.2->pdpbox==0.2.0) (23.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.8/dist-packages (from matplotlib>=2.1.2->pdpbox==0.2.0) (7.1.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib>=2.1.2->pdpbox==0.2.0) (1.0.7)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.8/dist-packages (from matplotlib>=2.1.2->pdpbox==0.2.0) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.8/dist-packages (from pandas->pdpbox==0.2.0) (2022.7.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.7->matplotlib>=2.1.2->pdpbox==0.2.0) (1.15.0)
Building wheels for collected packages: pdpbox
  Building wheel for pdpbox (setup.py) ... done
  Created wheel for pdpbox: filename=PDPbox-0.2.0-py3-none-any.whl size=57690718 sha256=32b17eb8b3ec8eb388ee7444d88497f0042d348653eccd57c956ec675a07bb53
  Stored in directory: /root/.cache/pip/wheels/7f/7d/f5/136844ad90a5cfb60b92f6de7afa6d8574042b6012a6d16e2b
Successfully built pdpbox
Installing collected packages: pdpbox
Successfully installed pdpbox-0.2.0
```

```
import math
import machine_learning_datasets as mldatasets
import pandas as pd
import numpy as np
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn import metrics, linear_model, tree, discriminant_analysis,\
                    ensemble, neural_network, inspection
import matplotlib.pyplot as plt
from pdpbox import pdp
from pycebox.ice import ice, ice_plot
```

```
/usr/local/lib/python3.8/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/base/tsa_model.py:8: FutureWarning: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.
  from pandas import (to_datetime, Int64Index, DatetimeIndex, Period,
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/base/tsa_model.py:8: FutureWarning: pandas.Float64Index is deprecated and will be removed from pandas in a future version. Use panda
  from pandas import (to_datetime, Int64Index, DatetimeIndex, Period,
```

```
pip install matplotlib==2.1.1
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: matplotlib==2.1.1 in /usr/local/lib/python3.8/dist-packages (2.1.1)
Requirement already satisfied: numpy>=1.7.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib==2.1.1) (1.22.4)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib==2.1.1) (0.11.0)
Requirement already satisfied: pytz in /usr/local/lib/python3.8/dist-packages (from matplotlib==2.1.1) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.0 in /usr/local/lib/python3.8/dist-packages (from matplotlib==2.1.1) (2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib==2.1.1) (3.0.9)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib==2.1.1) (1.15.0)
```

```
birthorder_df = mldatasets.load("personality-birthorder", prepare=True)
```

```
https://openpsychometrics.org/_rawdata/FBPS-ValidationData.zip downloaded to /content/data/FBPS-ValidationData.zip
/content/data/FBPS-ValidationData.zip uncompressed to /content/data/FBPS-ValidationData
1 dataset files found in /content/data/FBPS-ValidationData folder
parsing /content/data/FBPS-ValidationData/FBPS-ValidationData/FBPS-ValidationData.csv
```

```
birthorder_df.shape
```

```
(25813, 97)
```

▾ Data Preparation

```
rand = 9
y = birthorder_df['birthorder']
X = birthorder_df.drop(['birthorder'], axis=1).copy()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=rand)
```

Classification Model

```
class_models = {
        #Tree
        'decision_tree':{'model': tree.\
                         DecisionTreeClassifier(max_depth=6, random_state=rand,\
                                                class_weight='balanced')},
        #Ensemble Methods
        'gradient_boosting':{'model':ensemble.\
                             GradientBoostingClassifier(n_estimators=200,\
                                                        max_depth=4, subsample=0.5,\
                                                        learning_rate=0.05)},
        'random_forest':{'model':ensemble.\
                         RandomForestClassifier(max_depth=11, n_estimators=300,\
                                                max_features='sqrt', random_state=rand)},
        #Generalized Linear Models (GLMs)
        'logistic':{'model': linear_model.\
                    LogisticRegression(multi_class='ovr', solver='lbfgs',\
                                       class_weight='balanced', max_iter=500)},
        #Discriminant Analysis
        'lda':{'model':discriminant_analysis.\
               LinearDiscriminantAnalysis(n_components=2)},
        #Neural Networks
        'mlp':{'model':make_pipeline(StandardScaler(), neural_network.\
                                     MLPClassifier(hidden_layer_sizes=(11,),\
                                                   early_stopping=True, random_state=rand,\
                                                   validation_fraction=0.25, max_iter=500))}
    }
```

```
for model_name in class_models.keys():
    fitted_model = class_models[model_name]['model'].fit(X_train, y_train)
    y_train_pred = fitted_model.predict(X_train)
    y_test_pred = fitted_model.predict(X_test)
    class_models[model_name]['fitted'] = fitted_model
    class_models[model_name]['preds'] = y_test_pred
    class_models[model_name]['Accuracy_train'] =\
        metrics.accuracy_score(y_train, y_train_pred)
    class_models[model_name]['Accuracy_test'] =\
        metrics.accuracy_score(y_test, y_test_pred)
    class_models[model_name]['Recall_train'] =\
        metrics.recall_score(y_train, y_train_pred, average='weighted')
    class_models[model_name]['Recall_test'] =\
        metrics.recall_score(y_test, y_test_pred, average='weighted')
    class_models[model_name]['Precision_train'] =\
        metrics.precision_score(y_train, y_train_pred, average='weighted')
    class_models[model_name]['Precision_test'] =\
        metrics.precision_score(y_test, y_test_pred, average='weighted')
    class_models[model_name]['F1_test'] =\
        metrics.f1_score(y_test, y_test_pred, average='weighted')
    class_models[model_name]['MCC_test'] =\
        metrics.matthews_corrcoef(y_test, y_test_pred)
```

```
pip install jinja2==3.0.0
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: jinja2==3.0.0 in /usr/local/lib/python3.8/dist-packages (3.0.0)
Requirement already satisfied: MarkupSafe>=2.0.0rc2 in /usr/local/lib/python3.8/dist-packages (from jinja2==3.0.0) (2.0.1)
```

```
import pandas as pd
```

```
class_metrics = pd.DataFrame.\
    from_dict(class_models, 'index')[['Accuracy_train', 'Accuracy_test',\
                                      'Recall_train', 'Recall_test',\
                                      'Precision_train', 'Precision_test',\
                                      'F1_test', 'MCC_test']]
with pd.option_context('display.precision', 3):
    html = class_metrics.sort_values(by='MCC_test', ascending=False).style.\
        background_gradient(cmap='plasma', low=0.43, high=0.63,\
                            subset=['Accuracy_train', 'Accuracy_test']).\
        background_gradient(cmap='viridis', low=0.63, high=0.43,
                            subset=['F1_test'])
html
```

	Accuracy_train	Accuracy_test	Recall_train	Recall_test	Precision_train	Precision_test	F1_test	MCC_test
decision_tree	0.496704	0.464139	0.496704	0.464139	0.541157	0.493572	0.441380	0.245647
logistic	0.498323	0.497094	0.498323	0.497594	0.499407	0.497730	0.495107	0.237932
gradient_boosting	0.625419	0.492663	0.625419	0.492663	0.638101	0.487673	0.479188	0.227185
mlp	0.522031	0.494424	0.522031	0.494424	0.517411	0.485165	0.480317	0.222783
lda	0.500810	0.492077	0.500810	0.492077	0.499672	0.489145	0.476930	0.201126
random_forest	0.911588	0.483625	0.911588	0.483625	0.920902	0.477702	0.446664	0.197593

```
print('NIR: %.4f' %\
      (y_train[y_train==1].shape[0]/y_train.shape[0]))
```

```
NIR: 0.4215
```

Feature Importance for Tree Based Model

```
dt_imp_df = pd.DataFrame({\
                'name': X_train.columns,
                'dt_imp': class_models['decision_tree']['fitted'].\
                          feature_importances_})
gb_imp_df = pd.DataFrame({\
                'name': X_train.columns,
                'gb_imp': class_models['gradient_boosting']['fitted'].\
                          feature_importances_})
rf_imp_df = pd.DataFrame({\
                'name': X_train.columns,
                'rf_imp': class_models['random_forest']['fitted'].\
                         feature_importances_})
```

```
dt_rank_df = pd.DataFrame({'dt_rank':\
                           dt_imp_df['dt_imp'].\
                           rank(method='first',\
                                ascending=False).astype(int)})
gb_rank_df = pd.DataFrame({'gb_rank':\
                           gb_imp_df['gb_imp'].\
                           rank(method='first',\
                                ascending=False).astype(int)})
rf_rank_df = pd.DataFrame({'rf_rank':\
                           rf_imp_df['rf_imp'].\
                           rank(method='first',\
                                ascending=False).astype(int)})
```

```
tree_ranks_df = pd.merge(\
                    pd.merge(\
                        pd.concat((dt_imp_df, dt_rank_df), axis=1),\
                        pd.concat((gb_imp_df, gb_rank_df), axis=1),\
                        'left'),\
                    pd.concat((rf_imp_df, rf_rank_df), axis=1),\
                    'left')
tree_ranks_df['avg_rank'] = (tree_ranks_df['dt_rank'] +\
                             tree_ranks_df['gb_rank'] + \
                             tree_ranks_df['rf_rank'])/3
tree_ranks_df.sort_values(by='avg_rank')
```

	name	dt_imp	dt_rank	gb_imp	gb_rank	rf_imp	rf_rank	avg_rank
28	birthn	0.851533	1	0.369384	1	0.196067	1	1.000000
81	testelapse	0.013708	3	0.035858	2	0.027307	2	2.333333
26	age	0.006679	7	0.031328	3	0.025065	3	4.333333
0	Q1	0.025340	2	0.025485	6	0.016135	6	4.666667
80	introelapse	0.005056	9	0.028393	4	0.022142	5	6.000000
...	...	...	...	...	...	...	...	...
93	source_undefined	0.000000	94	0.001321	89	0.003050	85	89.333333
86	gender_other	0.000000	88	0.000935	92	0.000678	94	91.333333
83	gender_undefined	0.000000	87	0.000998	91	0.000241	96	91.333333
91	country_NZ	0.000000	92	0.000698	95	0.000690	93	93.333333
90	country_IE	0.000000	91	0.000630	96	0.000577	95	94.000000

96 rows × 8 columns

```
print(class_models['logistic']['fitted'].coef_.shape)
```

```
(3, 96)
```

```
stdv = np.std(X_train, 0)
lr_imp_df = pd.DataFrame({\
                'name': X_train.columns,\
                'first_coef_norm':\
                          class_models['logistic']['fitted'].coef_[0] * stdv,\
                'middle_coef_norm':\
                          class_models['logistic']['fitted'].coef_[1] * stdv,\
                'last_coef_norm':\
                          class_models['logistic']['fitted'].coef_[2] * stdv}).\
            reset_index(drop=True)
```

```
class_priors = class_models['lda']['fitted'].priors_
print(class_priors)
```

```
[0.42147566 0.23701862 0.34150572]
```

```
lr_imp_df['coef_weighted_avg'] = (abs(lr_imp_df['first_coef_norm']) * class_priors[0]) +\
                                 (abs(lr_imp_df['middle_coef_norm']) * class_priors[1]) +\
                                 (abs(lr_imp_df['last_coef_norm']) * class_priors[2])
```

```
lr_imp_df.\
    sort_values(by='coef_weighted_avg', ascending=False).style.\
    background_gradient(cmap='viridis', low=-0.1, high=0.1,\
                        subset=['first_coef_norm',\
                                'middle_coef_norm', 'last_coef_norm'])
```

	name	first_coef_norm	middle_coef_norm	last_coef_norm	coef_weighted_avg
28	birthn	-0.486102	1.274109	-0.406319	0.645628
0	Q1	0.123289	0.023123	-0.149011	0.108332
12	Q13	0.085769	-0.048447	-0.063696	0.069385
15	Q16	0.063228	-0.057450	-0.027473	0.049648
19	Q20	-0.058521	0.071115	0.022708	0.049276
1	Q2	0.010945	-0.099632	0.048793	0.044891
9	Q10	0.052110	-0.017730	-0.045738	0.041785
51	AGR3	-0.042034	-0.024955	0.049640	0.040583
26	age	0.032169	-0.066691	-0.029110	0.039306
39	EST1	0.042110	-0.085655	-0.002980	0.039068
37	EXT9	0.028834	-0.067813	0.027615	0.037656
18	Q19	0.044459	-0.044302	-0.022392	0.036886
3	Q4	0.045280	-0.062586	-0.005421	0.035770
40	EST2	-0.035091	0.028893	0.039463	0.035115
25	Q26	0.045389	-0.004068	-0.042252	0.034524
24	Q25	-0.045834	0.035025	0.019754	0.034365
59	CSN1	0.040003	-0.063954	0.003836	0.033329
16	Q17	0.037205	0.008680	-0.042894	0.032387
77	OPN9	-0.040012	0.016503	0.028996	0.030678
82	endelapse	-0.045790	0.010989	0.024235	0.030181

```
lr_imp_df.\
    sort_values(by='coef_weighted_avg', ascending=False).style.\
    background_gradient(cmap='viridis', low=-0.1, high=0.1,\
                        subset=['first_coef_norm',\
                                'middle_coef_norm', 'last_coef_norm'])
```

47	EST9	0.021887	-0.075068	0.009244	0.030174
71	OPN3	-0.037127	0.008642	0.035529	0.029830
11	Q12	0.035605	-0.038845	-0.011301	0.028073
41	EST3	0.025194	-0.063828	0.006735	0.028047
4	Q5	-0.027144	-0.011987	0.037567	0.027111
10	Q11	0.029573	-0.014530	-0.027757	0.025388
48	EST10	-0.025626	-0.027464	0.022747	0.025078
5	Q6	0.010719	0.020943	-0.039721	0.023047
68	CSN10	-0.018111	-0.020779	0.030038	0.022817
64	CSN6	0.018267	-0.044440	0.012810	0.022607
13	Q14	0.024738	-0.036380	-0.009849	0.022413
67	CSN9	0.027882	-0.035731	-0.006204	0.022339
74	OPN6	0.027231	-0.020809	-0.017057	0.022235
49	AGR1	0.028080	-0.025618	-0.011948	0.021987
50	AGR2	0.005056	-0.036231	0.029224	0.020699
21	Q22	0.002305	-0.043938	0.026064	0.020287
31	EXT3	0.034374	-0.020997	-0.000735	0.019716
14	Q15	0.010099	-0.041565	0.015774	0.019495
29	EXT1	-0.007013	-0.021009	0.031946	0.018845
81	testelapse	0.016831	-0.037885	-0.007577	0.018661
44	EST6	0.006882	-0.054720	0.006688	0.018154

Feature Importance for LDA:

```
class_models['lda']['fitted'].coef_.shape
```

```
(3, 96)
```

```
lda_imp_df = pd.DataFrame({\
                'name': X_train.columns,\
                'first_coef_norm':\
                          class_models['lda']['fitted'].coef_[0] * stdv,\
                'middle_coef_norm':\
                          class_models['lda']['fitted'].coef_[1] * stdv,\
                'last_coef_norm':\
                          class_models['lda']['fitted'].coef_[2] * stdv}).\
            reset_index(drop=True)
```

```
lda_imp_df['coef_weighted_avg'] = (abs(lda_imp_df['first_coef_norm']) * class_priors[0]) +\
                                  (abs(lda_imp_df['middle_coef_norm']) * class_priors[1]) +\
                                  (abs(lda_imp_df['last_coef_norm']) * class_priors[2])
```

```
lda_imp_df.\
    sort_values(by='coef_weighted_avg', ascending=False).style.\
    background_gradient(cmap='viridis', low=-0.1, high=0.1,\
                        subset=['first_coef_norm',\
                                'middle_coef_norm', 'last_coef_norm'])
```

```
lda_imp_df.\
    sort_values(by='coef_weighted_avg', ascending=False).style.\
    background_gradient(cmap='viridis', low=-0.1, high=0.1,\
                        subset=['first_coef_norm',\
                                'middle_coef_norm', 'last_coef_norm'])
```

	name	first_coef_norm	middle_coef_norm	last_coef_norm	coef_weighted_avg
28	birthn	-0.315051	1.002922	-0.307242	0.475423
0	Q1	0.090613	-0.012808	-0.102942	0.076382
12	Q13	0.056740	-0.033932	-0.046477	0.047829
51	AGR3	-0.039195	-0.005623	0.052276	0.035705
15	Q16	0.038815	-0.035811	-0.023050	0.032719
6	Q7	-0.004032	0.064381	-0.039707	0.030519
16	Q17	0.035050	-0.009894	-0.036390	0.029545
24	Q25	-0.034672	0.034036	0.019169	0.029227
77	OPN9	-0.033481	0.045381	0.009825	0.028223
19	Q20	-0.030615	0.038022	0.011395	0.025807
31	EXT3	0.030346	-0.001682	-0.036285	0.025580
9	Q10	0.028820	-0.011171	-0.027815	0.024293
3	Q4	0.028030	-0.028287	-0.014962	0.023628
47	EST9	0.017339	-0.049699	0.013094	0.023559
76	OPN8	0.022549	-0.048056	0.005523	0.022780
71	OPN3	-0.026843	0.030238	0.012143	0.022628
64	CSN6	0.019719	-0.047238	0.008448	0.022392
39	EST1	0.025820	-0.042403	-0.002437	0.021765
60	CSN2	-0.025170	0.040077	0.003249	0.021217
68	CSN10	-0.025030	0.036075	0.026050	0.021090

Feature Importance for MLP:

```
print(class_models['mlp']['fitted'][1].coefs_[0].shape)
print(class_models['mlp']['fitted'][1].coefs_[1].shape)
```

```
(96, 11)
(11, 3)
```

```
for model_name in class_models.keys():
    fitted_model = class_models[model_name]['fitted']
    permutation_imp = inspection.permutation_importance(\
                        fitted_model, X_test, y_test, n_jobs=-1,\
                        scoring='accuracy', n_repeats=8,\
                        random_state=rand)
    class_models[model_name]['importances_mean'] =\
                        permutation_imp.importances_mean
```

```
perm_imp_df = pd.DataFrame({
                'name': X_train.columns,\
                'dt_imp': class_models['decision_tree']['importances_mean'],\
                'gb_imp': class_models['gradient_boosting']['importances_mean'],\
                'rf_imp': class_models['random_forest']['importances_mean'],\
                'log_imp': class_models['logistic']['importances_mean'],\
                'lda_imp': class_models['lda']['importances_mean'],\
                'mlp_imp': class_models['mlp']['importances_mean']}).\
            reset_index(drop=True)
```

```
perm_imp_df['avg_imp'] = (perm_imp_df['dt_imp'] + perm_imp_df['gb_imp'] +\
                          perm_imp_df['rf_imp'] + perm_imp_df['log_imp'] +\
                          perm_imp_df['lda_imp'] + perm_imp_df['mlp_imp'])/6
```

```
perm_imp_sorted_df = perm_imp_df.round(5).\
    sort_values(by='avg_imp', ascending=False)
perm_imp_sorted_df.style.\
    background_gradient(cmap='viridis_r', low=0, high=0.2,\
                        subset=['dt_imp', 'gb_imp', 'rf_imp',\
                                'log_imp', 'lda_imp', 'mlp_imp'])
```

	name	dt_imp	gb_imp	rf_imp	log_imp	lda_imp	mlp_imp	avg_imp
28	birthn	0.138500	0.107270	0.074610	0.115140	0.083300	0.109310	0.104690
0	Q1	0.008320	0.002360	0.000880	0.011420	0.011400	0.008230	0.007100
12	Q13	0.000980	-0.003360	0.001670	0.003150	0.004270	0.002830	0.001590
26	age	0.001070	0.000850	0.004110	0.001690	-0.000820	0.001580	0.001410
51	AGR3	0.000320	-0.002540	0.000810	0.003050	0.003100	0.002950	0.001280
3	Q4	0.000000	-0.001610	0.002050	0.001420	0.000900	0.002820	0.000930
63	CSN5	0.002040	-0.000750	0.000130	0.000840	0.000920	0.001530	0.000790
4	Q5	-0.000790	-0.000320	-0.000190	0.002330	0.001250	0.001750	0.000670
14	Q15	0.000000	-0.000210	0.001980	0.000120	0.000760	0.001230	0.000650
24	Q25	0.000000	-0.001970	0.000230	0.000900	0.002050	0.002390	0.000600
32	EXT4	0.000000	0.001140	0.000730	0.000440	-0.000250	0.001350	0.000570
59	CSN1	0.000000	-0.001730	0.001350	0.000760	0.000030	0.002860	0.000550
9	Q10	0.002820	-0.001090	0.000250	0.001500	0.000400	-0.000630	0.000540
13	Q14	0.003210	-0.001190	-0.001190	-0.000160	0.000380	0.001880	0.000490
52	AGR4	0.000000	-0.000840	0.000470	-0.000090	0.000470	0.002950	0.000490
16	Q17	0.001190	-0.002920	-0.000040	0.001660	0.001800	0.001060	0.000460
84	gender_male	0.002700	-0.000150	-0.000290	0.000560	0.000530	-0.000910	0.000410
50	AGR2	0.000000	-0.000130	0.000600	-0.000070	0.000660	0.001410	0.000410
87	country_AU	0.000000	0.000660	0.000530	-0.000000	0.000290	0.000900	0.000400

```
pd.DataFrame.\
    from_dict(class_models, 'index')[['Accuracy_test']] -\
    perm_imp_sorted_df.iloc[0,1:7].to_numpy().reshape((6,1))
```

	Accuracy_test
decision_tree	0.325639
gradient_boosting	0.385393
random_forest	0.409015
logistic	0.382454
lda	0.408777
mlp	0.385114

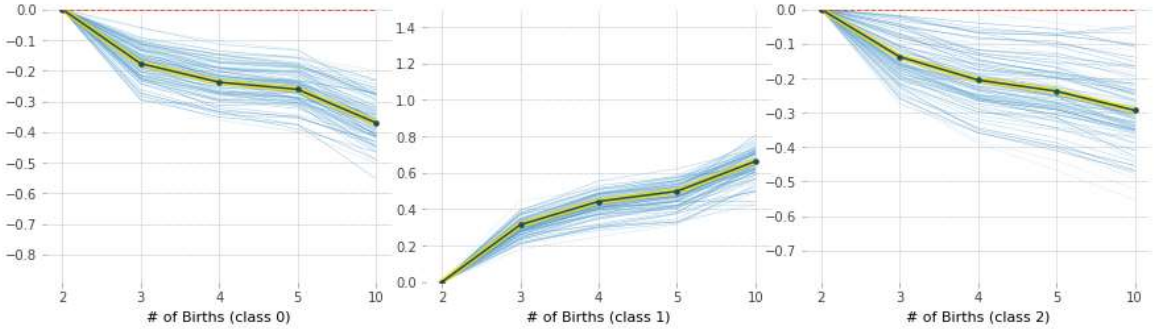
Partial dependence Plots

```
feature_names = ['birthn', 'Q1', 'Q13', 'age']
feature_labels = ['# of Births', 'Question #1',\
                  'Question #13', 'Age']
```

```
for i in range(len(feature_names)):
    pdp_feat_df = pdp.pdp_isolate(\
        model=class_models['gradient_boosting']['fitted'],\
        dataset=pd.concat((X_test, y_test), axis=1),\
        model_features=X_test.columns,\
        feature=feature_names[i]
    )
    fig, axes = pdp.pdp_plot(
        pdp_isolate_out=pdp_feat_df, center=True,\
        x_quantile=True, ncols=3, plot_lines=True,\
        frac_to_plot=100, figsize=(15,6),\
        feature_name=feature_labels[i]
    )
```

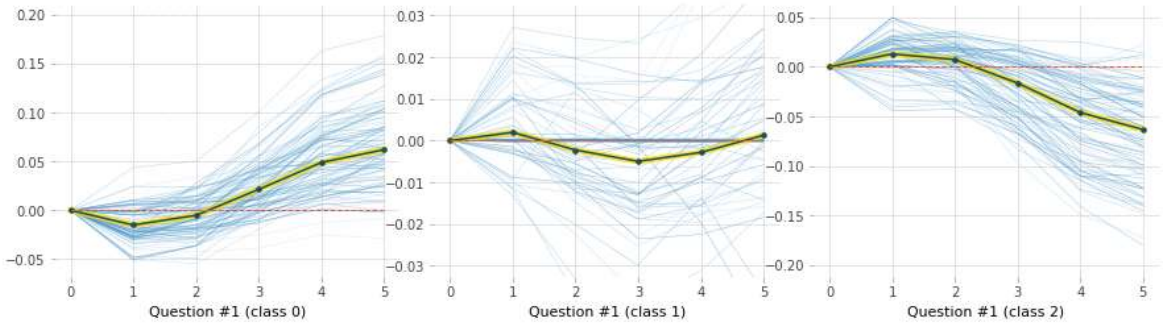
PDP for feature "# of Births"

Number of unique grid points: 5



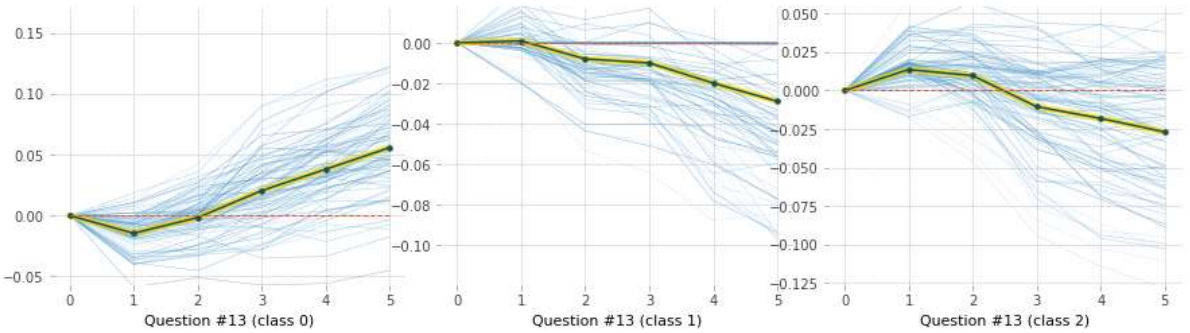
PDP for feature "Question #1"

Number of unique grid points: 6



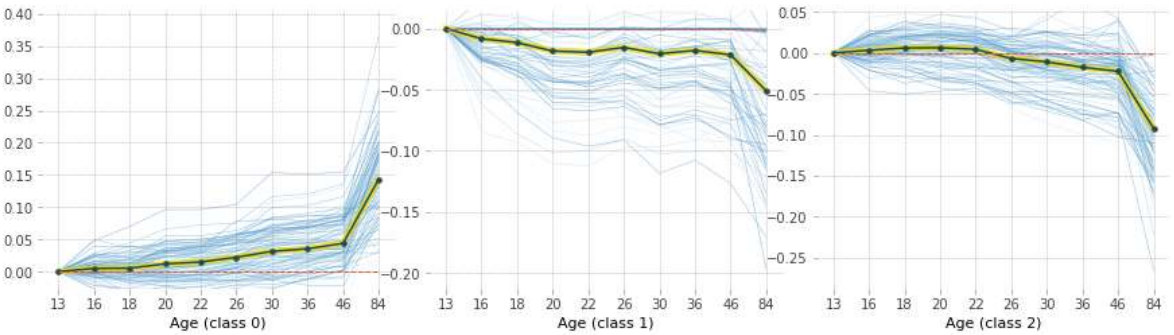
PDP for feature "Question #13"

Number of unique grid points: 6



PDP for feature "Age"

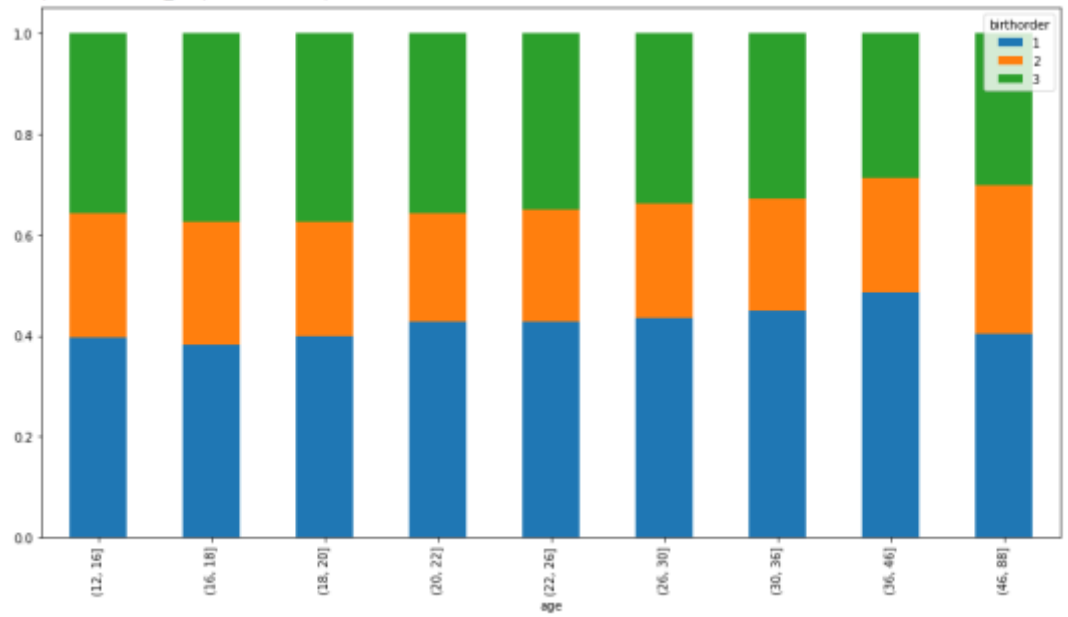
Number of unique grid points: 10




```
birthorder_abbrev_df = birthorder_df[['age', 'birthorder']]
birthorder_abbrev_df.set_index(pd.cut(\
                                birthorder_abbrev_df['age'],\
                                [12, 16, 18, 20, 22, 26, 30,\
                                 36, 46, 88]), inplace=True)
agegroup_birthorder_counts_s = birthorder_abbrev_df.\
                           groupby([birthorder_abbrev_df.index,\
                                    'birthorder']).size()
agegroup_counts_s = birthorder_abbrev_df.groupby(\
                                          birthorder_abbrev_df.index)\
                                          ['birthorder'].count()
agegroup_pct_birthorder_s = agegroup_birthorder_counts_s.div(\
                                                    agegroup_counts_s,\
                                                     axis=0,level=0)
agegroup_pct_birthorder_s.unstack().plot.bar(stacked=True,\
                                        figsize=(15,8))
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fefd9583240>

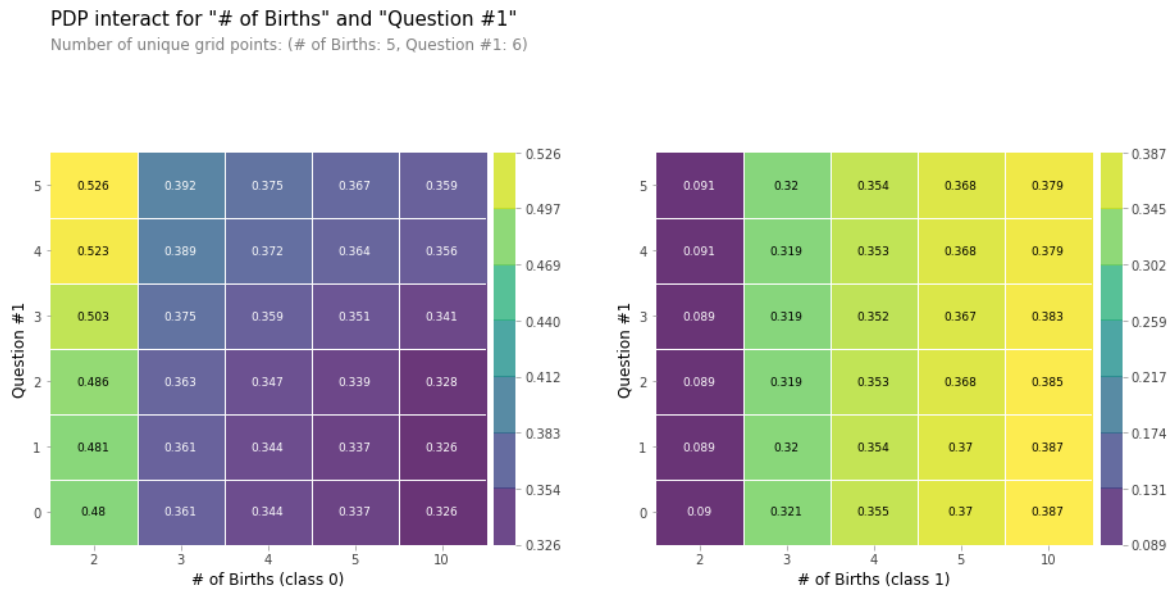
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fefd9583340>
```

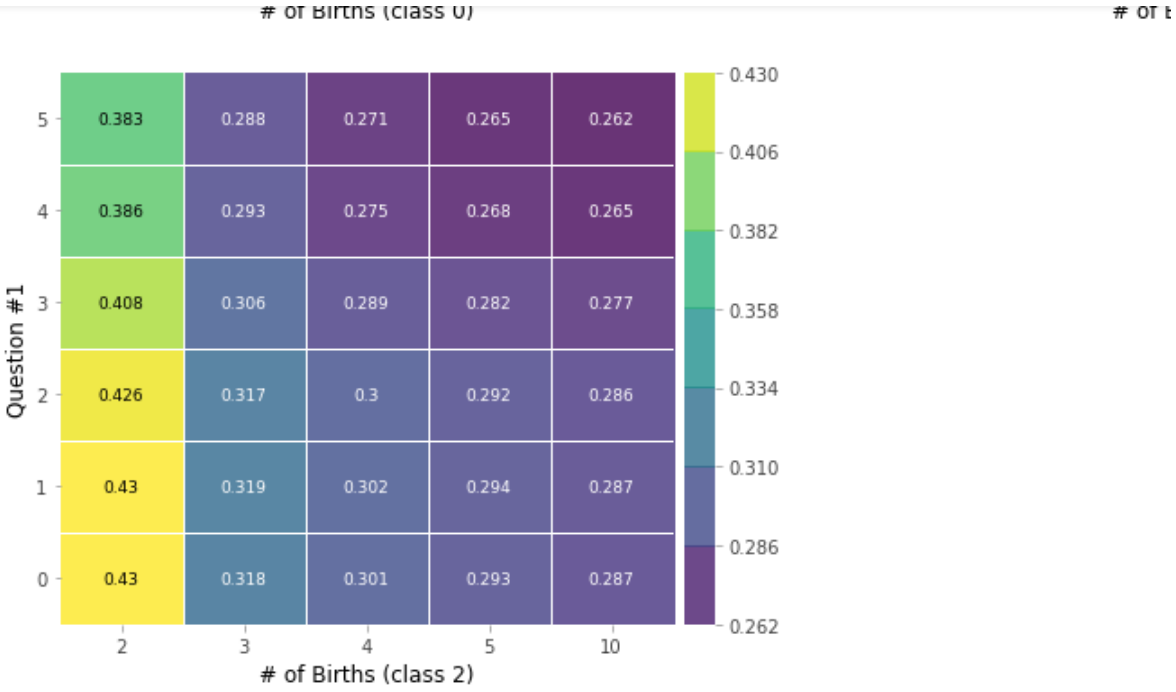


PDPs

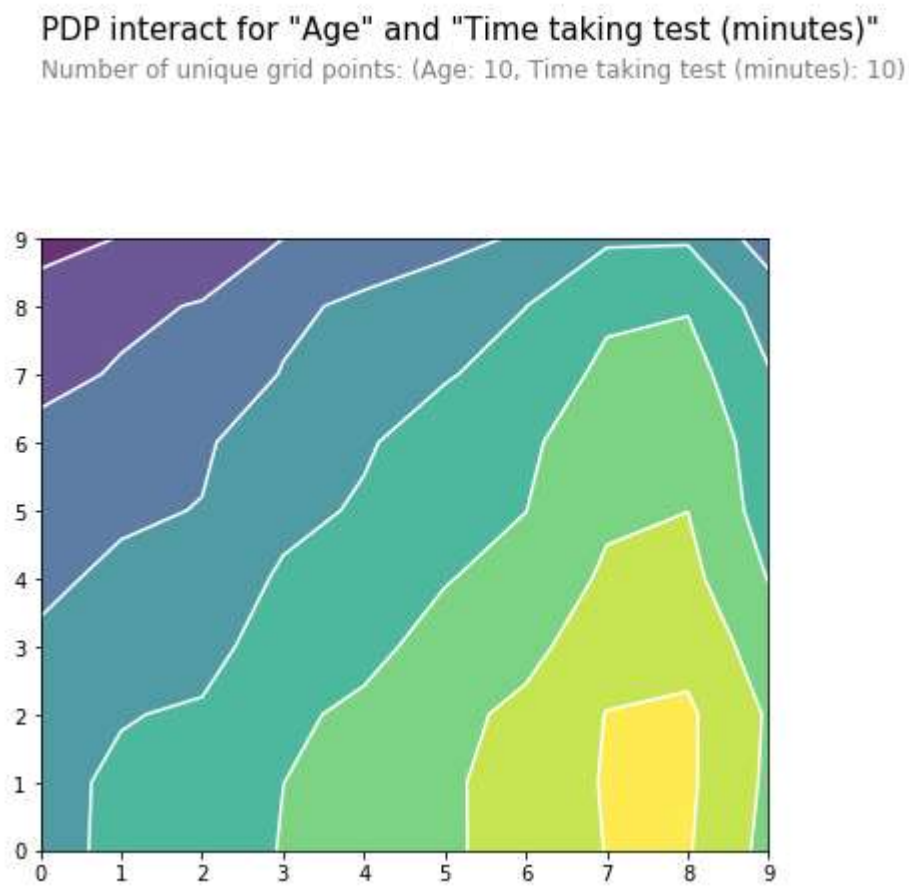
```
pdp_birthn_Q1_df = pdp.pdp_interact(
    model=class_models['random_forest']['fitted'],\
    dataset=pd.concat((X_test, y_test), axis=1),\
    model_features=X_test.columns, features=['birthn','Q1'],\
    n_jobs=-1
)
fig, axes = pdp.pdp_interact_plot(
    pdp_interact_out=pdp_birthn_Q1_df,\
    plot_type='grid', x_quantile=True,\
    ncols=2, figsize=(15,15),\
    feature_names=['# of Births','Question #1']
)
```

```
WARNING:matplotlib.font_manager:findfont: Font family ['Arial'] not found. Falling back to DejaVu Sans.
```






```
pdp_age_testelapse_df = pdp.pdp_interact(
    model=class_models['random_forest']['fitted'],\
    dataset=pd.concat((X_test, y_test), axis=1),\
    model_features=X_test.columns, features=['age','testelapse'],\
    n_jobs=-1
)
fig, axes = pdp.pdp_interact_plot(
    pdp_interact_out=pdp_age_testelapse_df,\
    plot_type='contour', x_quantile=True,\
    ncols=2, figsize=(15,15),\
    feature_names=['Age','Time taking test (minutes)']
)
```


PDP interact for "Age" and "Time taking test (minutes)"
Number of unique grid points: (Age: 10, Time taking test (minutes): 10)
0
1
2
3
4
5
6
7
8
9

ICE Plots:

```
np.random.seed(rand)
sample_size = 0.1
sample_idx = np.random.choice(\
                    X_test.shape[0],\
                    math.ceil(X_test.shape[0]*sample_size),\
                    replace=False)
X_test_samp = X_test.iloc[sample_idx,:]
```

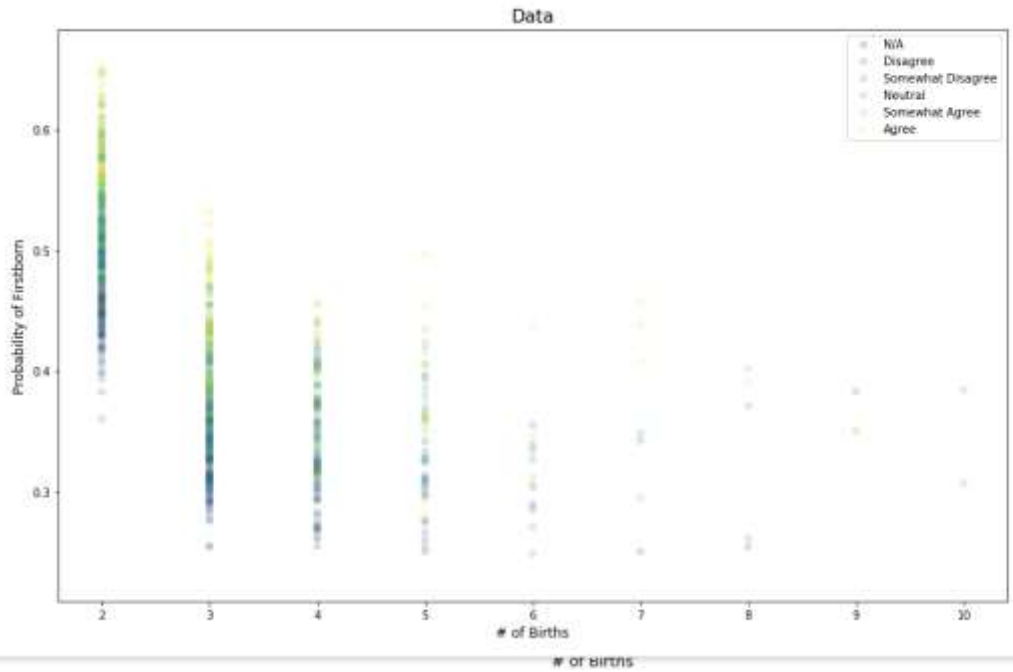
```
def predict_prob_first_born(test_df):
    return class_models['random_forest']['fitted'].\
                                predict_proba(test_df)[:,0]
def predict_prob_middle_child(test_df):
    return class_models['random_forest']['fitted'].\
                                predict_proba(test_df)[:,1]
def predict_prob_last_born(test_df):
    return class_models['random_forest']['fitted'].\
                                predict_proba(test_df)[:,2]
```

```
legend_key = {0:'N/A', 1:'Disagree', 2:'Somewhat Disagree',
              3:'Neutral', 4:'Somewhat Agree', 5:'Agree'}
```

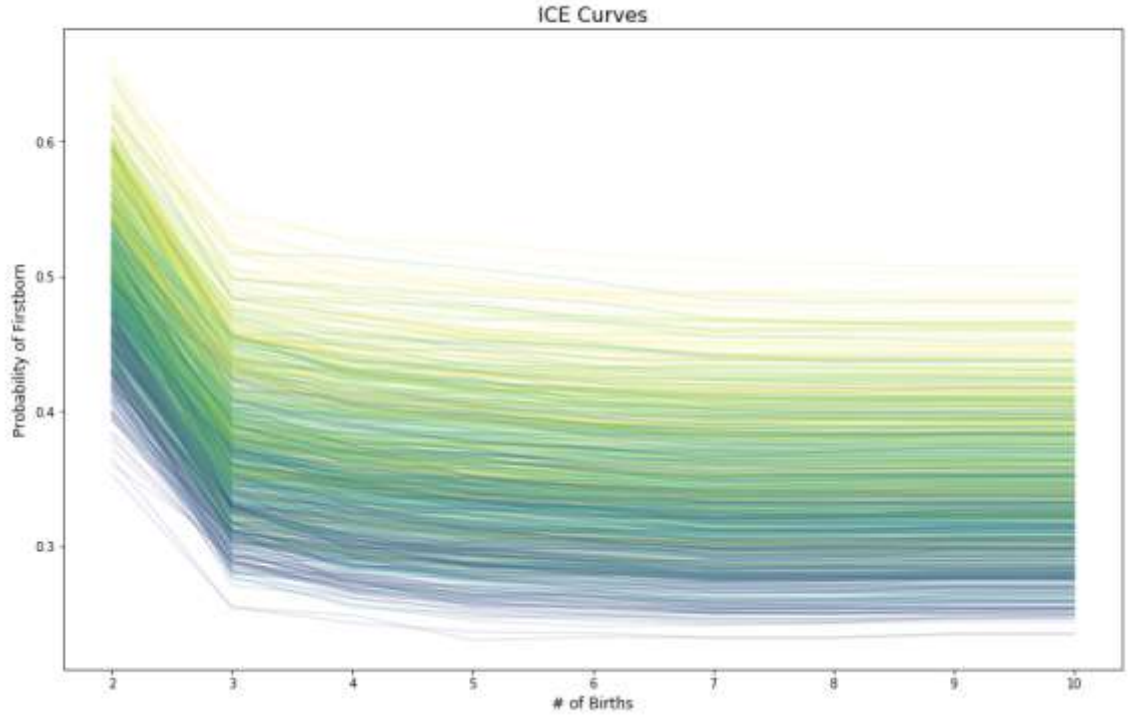
```
pip install matplotlib==3.1.2
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: matplotlib==3.1.2 in /usr/local/lib/python3.8/dist-packages (3.1.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.1.2) (2.8.2)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.1.2) (1.22.4)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.1.2) (3.0.9)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.1.2) (1.4.4)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.1.2) (0.11.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.1->matplotlib==3.1.2) (1.15.0)
```

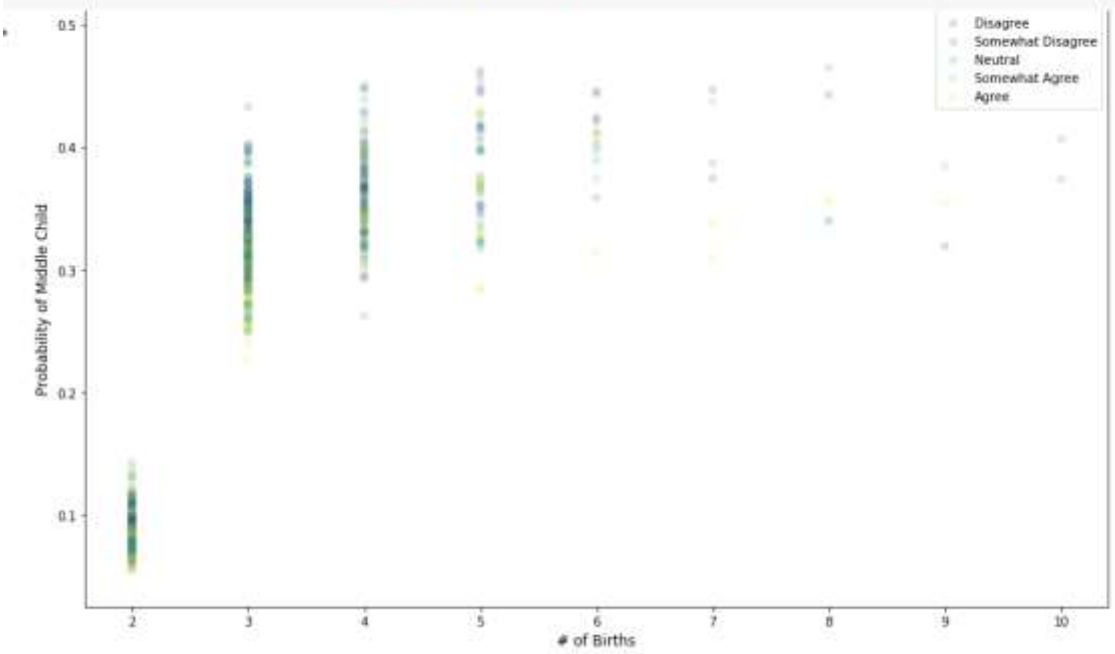
```
mldatasets.plot_data_vs_ice(predict_prob_first_born,\
                            'Probability of Firstborn',\
                            X=X_test_samp, feature_name='birthn',\
                            feature_label='# of Births',\
                            color_by='Q1', legend_key=legend_key)
```

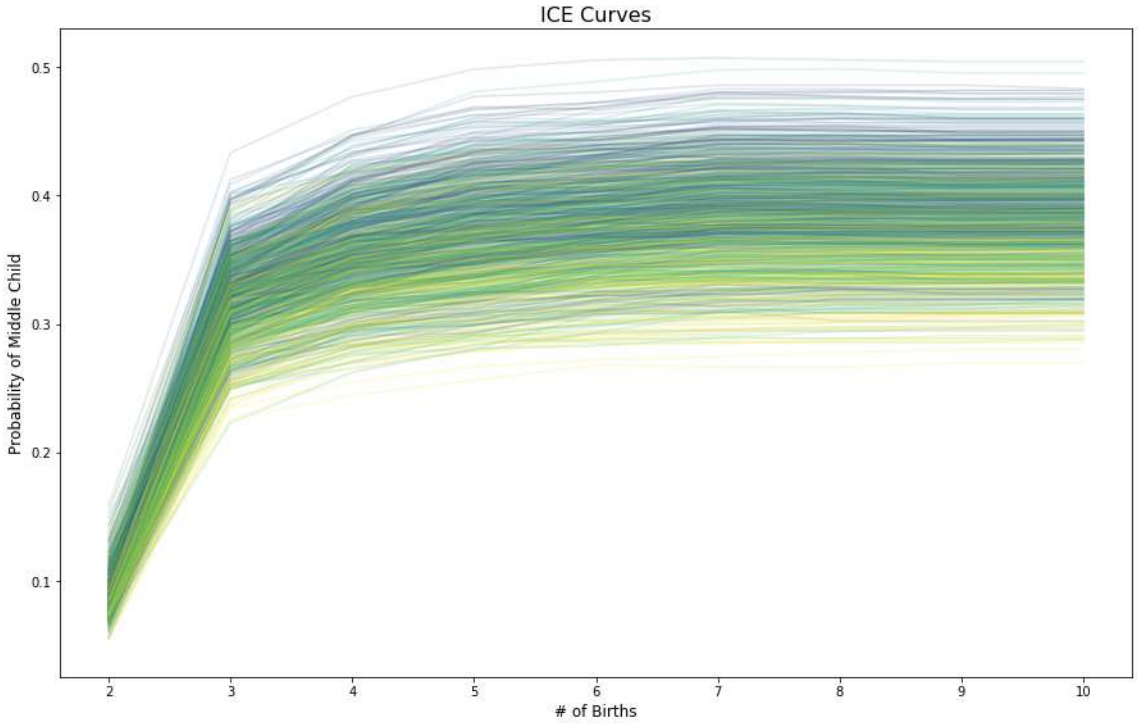

Data
N/A
Disagree
Somewhat Disagree
Neutral
Somewhat Agree
Agree
Probability of Firstborn
0.6
0.5
0.4
0.3
2
3
4
5
6
7
8
9
10
# of Births

# of Births


ICE Curves
Probability of Firstborn
0.6
0.5
0.4
0.3
2
3
4
5
6
7
8
9
10
# of Births

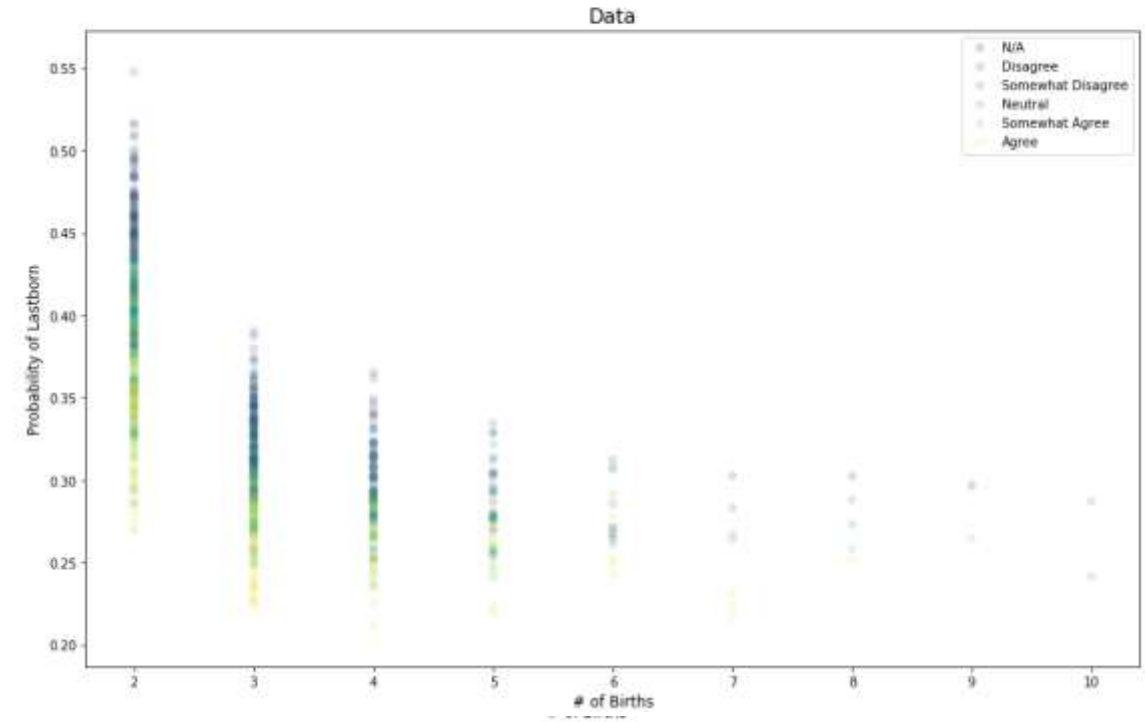
```
mldatasets.plot_data_vs_ice(predict_prob_middle_child,\
                            'Probability of Middle Child',\
                            X=X_test_samp, feature_name='birthn',\
                            feature_label='# of Births',\
                            color_by='Q1', legend_key=legend_key)
```

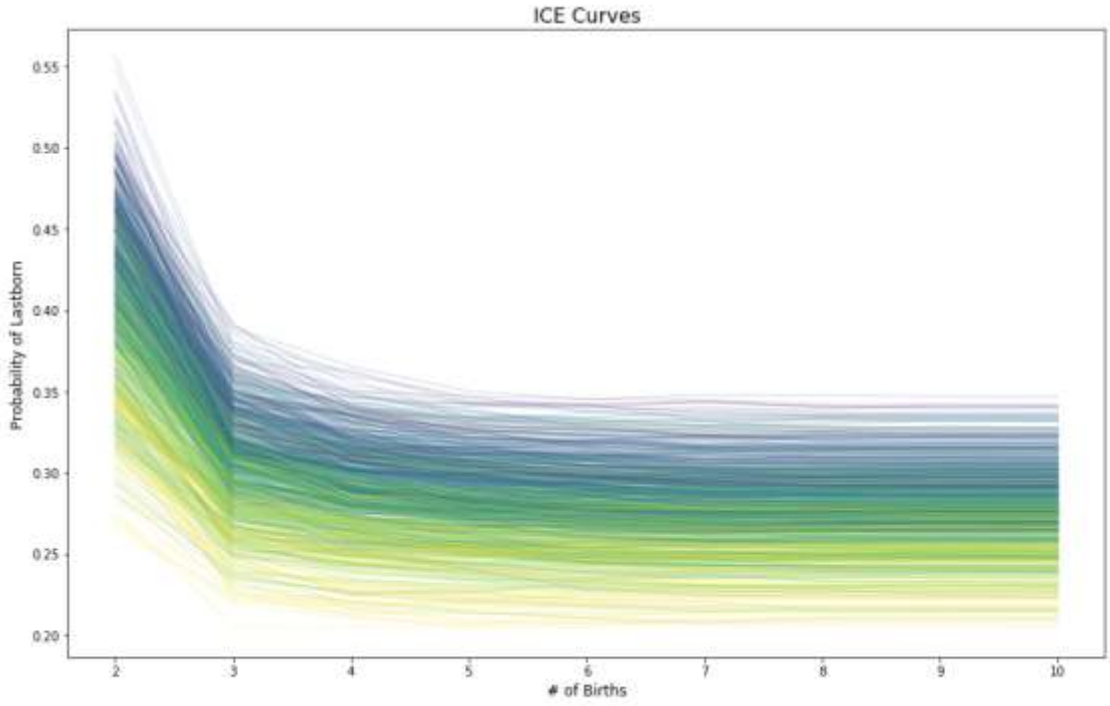





```
mldatasets.plot_data_vs_ice(predict_prob_last_born,\
                            'Probability of Lastborn',\
                            X=X_test_samp, feature_name='birthn',\
                            feature_label='# of Births',\
                            color_by='Q1', legend_key=legend_key)
```

```
mldatasets.plot_data_vs_ice(predict_prob_last_born,\
                            'Probability of Lastborn',\
                            X=X_test_samp, feature_name='birthn',\
                            feature_label='# of Births',\
                            color_by='Q1', legend_key=legend_key)
```





Chapter 5 (Fuel Efficiency):

```
!pip install --upgrade pandas numpy scikit-learn tensorflow xgboost scipy matplotlib seaborn
```

```
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.8/dist-packages (from markdown>=2.6.8->tensorboard<2.12,>=2.11->tensorflow) (6.0.0)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (1.24.3)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.8/dist-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.12,>=2.11->tensorflow) (3.12.0)
Requirement already satisfied: pyasn1>=0.5.0,<0.6.0 in /usr/local/lib/python3.8/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.8/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tensorflow
Installing collected packages: flatbuffers, tensorflow-estimator, numpy, keras, fonttools, scipy, pandas, contourpy, xgboost, scikit-learn, matplotlib, tensorboard, seaborn, tensorflow
  Attempting uninstall: flatbuffers
    Found existing installation: flatbuffers 1.12
    Uninstalling flatbuffers-1.12:
      Successfully uninstalled flatbuffers-1.12
  Attempting uninstall: tensorflow-estimator
    Found existing installation: tensorflow-estimator 2.9.0
    Uninstalling tensorflow-estimator-2.9.0:
      Successfully uninstalled tensorflow-estimator-2.9.0
  Attempting uninstall: numpy
    Found existing installation: numpy 1.21.6
    Uninstalling numpy-1.21.6:
      Successfully uninstalled numpy-1.21.6
  Attempting uninstall: keras
    Found existing installation: keras 2.9.0
    Uninstalling keras-2.9.0:
      Successfully uninstalled keras-2.9.0
  Attempting uninstall: scipy
    Found existing installation: scipy 1.7.3
```

```
!pip install --upgrade machine-learning-datasets
!pip install git+https://github.com/tensorflow/docs
!pip install --upgrade rulefit shap
!pip install git+https://github.com/MaximeJumelle/ALEPython.git@dev#egg=alepython
```

```
WARNING: Ignoring invalid distribution -ensorflow (/usr/local/lib/python3.8/dist-packages)
WARNING: Ignoring invalid distribution -ensorflow (/usr/local/lib/python3.8/dist-packages)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting machine-learning-datasets
  Downloading machine_learning_datasets-0.1.16.4-py3-none-any.whl (25 kB)
Collecting pycebox<0.0.2,>=0.0.1
  Downloading pycebox-0.0.1.tar.gz (4.5 kB)
  Preparing metadata (setup.py) ... done
Collecting seaborn<0.12.0,>=0.11.1
  Downloading seaborn-0.11.2-py3-none-any.whl (292 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 292.8/292.8 KB 7.9 MB/s eta 0:00:00
Collecting pathlib2<3.0.0,>=2.3.5
  Downloading pathlib2-2.3.7.post1-py2.py3-none-any.whl (18 kB)
Requirement already satisfied: matplotlib<4.0.0,>=3.2.2 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (3.6.3)
Requirement already satisfied: scipy<2.0.0,>=1.4.1 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (1.10.0)
Requirement already satisfied: pandas<2.0.0,>=1.1.5 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (1.5.3)
Collecting scikit-learn<0.23.0,>=0.22.2.post1
  Downloading scikit_learn-0.22.2.post1-cp38-cp38-manylinux1_x86_64.whl (7.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.0/7.0 MB 22.2 MB/s eta 0:00:00
Requirement already satisfied: opencv-python<5.0.0,>=4.5.1 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (4.6.0.66)
Requirement already satisfied: tqdm<5.0.0,>=4.41.1 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (4.64.1)
Requirement already satisfied: numpy<2.0.0,>=1.19.5 in /usr/local/lib/python3.8/dist-packages (from machine-learning-datasets) (1.24.2)
Collecting alibi<0.6.0,>=0.5.5
  Downloading alibi-0.5.8-py3-none-any.whl (312 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 312.5/312.5 KB 15.0 MB/s eta 0:00:00
```

```
import math
import os
import machine_learning_datasets as mldatasets
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import metrics, tree
import tensorflow as tf
import tensorflow_docs as tfdocs
import tensorflow_docs.plots
import xgboost as xgb
from rulefit import RuleFit
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
import shap
from alepython import ale_plot
```

```
print(tf.__version__)
```

```
2.2.1
```

```
fueleconomy_df = mldatasets.load("vehicle-fueleconomy", prepare=True)
```

```
https://www.fueleconomy.gov/feg/epadata/vehicles.csv.zip downloaded to /Users/smasis/Documents/OTHER/InterpretableMLBook/programming/Chapter5/data
/Users/smasis/Documents/OTHER/InterpretableMLBook/programming/Chapter5/data/vehicles.csv.zip uncompressed to /Users/smasis/Documents/OTHER/Interpr
1 dataset files found in /Users/smasis/Documents/OTHER/InterpretableMLBook/programming/Chapter5/data/vehicles.csv folder
parsing /Users/smasis/Documents/OTHER/InterpretableMLBook/programming/Chapter5/data/vehicles.csv/vehicles.csv
```

```
fueleconomy_df.info()
```

```
 28  drive_Part-time_4-Wheel                  43317 non-null  uint8
 29  drive_Rear-Wheel                         43317 non-null  uint8
 30  drive_Unknown                            43317 non-null  uint8
 31  VClass_Compact_Cars                      43317 non-null  uint8
 32  VClass_Large_Cars                        43317 non-null  uint8
 33  VClass_Midsize-Large_Station_Wagons      43317 non-null  uint8
 34  VClass_Midsize_Cars                      43317 non-null  uint8
 35  VClass_Midsize_Station_Wagons            43317 non-null  uint8
 36  VClass_Minicompact_Cars                  43317 non-null  uint8
 37  VClass_Other                             43317 non-null  uint8
 38  VClass_Small_Pickup_Trucks               43317 non-null  uint8
 39  VClass_Small_Sport_Utility_Vehicle_2WD   43317 non-null  uint8
 40  VClass_Small_Sport_Utility_Vehicle_4WD   43317 non-null  uint8
 41  VClass_Small_Station_Wagons              43317 non-null  uint8
 42  VClass_Special_Purpose_Vehicle_2WD       43317 non-null  uint8
 43  VClass_Special_Purpose_Vehicles          43317 non-null  uint8
 44  VClass_Sport_Utility_Vehicle_2WD         43317 non-null  uint8
 45  VClass_Sport_Utility_Vehicle_4WD         43317 non-null  uint8
 46  VClass_Standard_Pickup_Trucks            43317 non-null  uint8
 47  VClass_Standard_Pickup_Trucks_2WD        43317 non-null  uint8
 48  VClass_Standard_Pickup_Trucks_4WD        43317 non-null  uint8
 49  VClass_Standard_Sport_Utility_Vehicle_4WD 43317 non-null  uint8
 50  VClass_Subcompact_Cars                   43317 non-null  uint8
 51  VClass_Two_Seaters                       43317 non-null  uint8
 52  VClass_Vans                              43317 non-null  uint8
```

```
rand = 9
y = fueleconomy_df['comb08']
X = fueleconomy_df.drop(['comb08','make','model'], axis=1).copy()
X_train, X_test, y_train, y_test = train_test_split(X, y,\
                                    test_size=0.15, random_state=rand)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,\
                                    test_size=0.2, random_state=rand)
```

Deep Neural Network

```
os.environ['PYTHONHASHSEED']=str(rand)
tf.random.set_seed(rand)
np.random.seed(rand)
```

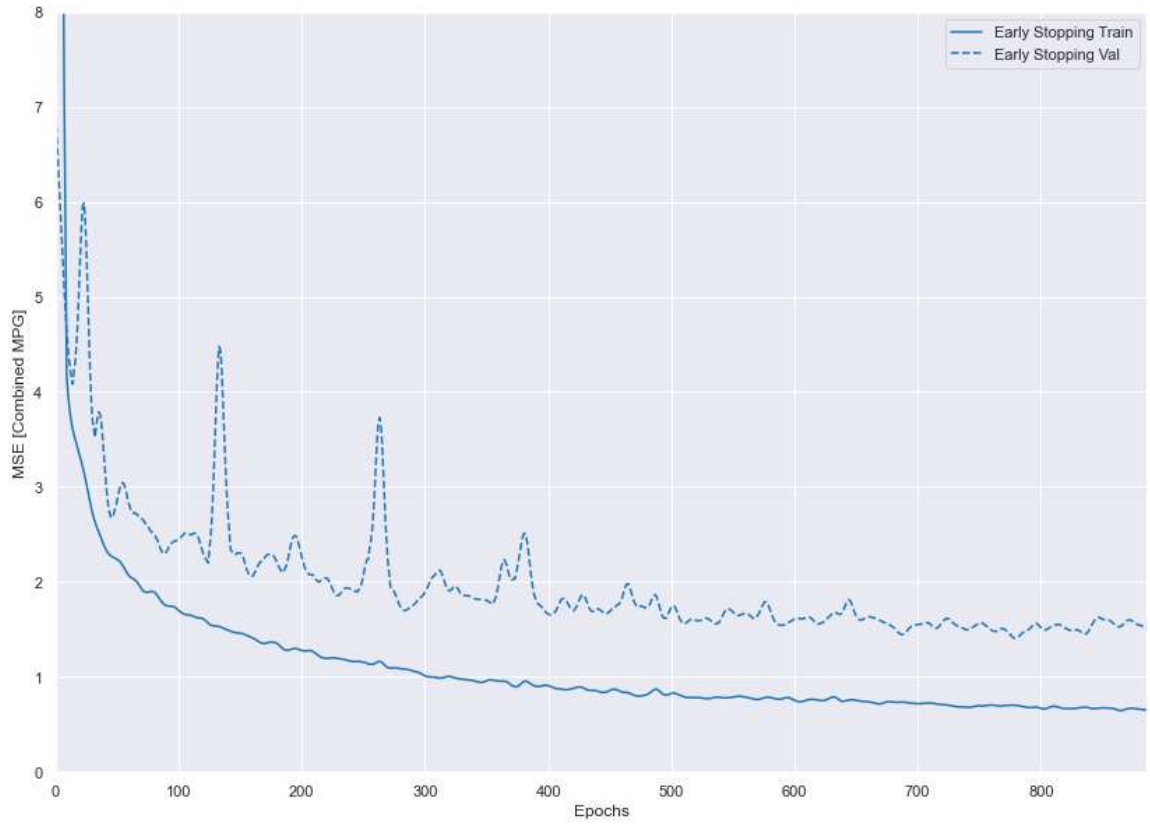
```
fitted_nn_model = tf.keras.Sequential([
    tf.keras.Input(shape=[len(X_train.keys())]),
    tf.keras.layers.experimental.preprocessing.Normalization(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1)
  ])
fitted_nn_model.compile(loss='mean_squared_error',\
                optimizer=tf.keras.optimizers.Adam(lr = 0.0005),\
                metrics=['mse'])
fitted_nn_model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
normalization_5 (Normalizati (None, 81)                163
_________________________________________________________________
dense_18 (Dense)             (None, 64)                5248
_________________________________________________________________
dense_19 (Dense)             (None, 64)                4160
_________________________________________________________________
dense_20 (Dense)             (None, 1)                 65
=================================================================
Total params: 9,636
Trainable params: 9,473
Non-trainable params: 163
_________________________________________________________________
```

```
es = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1,\
                                      patience=200, min_delta=0.0005,\
                                      restore_best_weights=True)
nn_history = fitted_nn_model.fit(\
                X_train.astype(float), y_train.astype(float), epochs=3000, batch_size=128,\
                validation_data=(X_val.values.astype(float), y_val.values.astype(float)), verbose=1,\
                                 callbacks=[es])
```

w hidden output

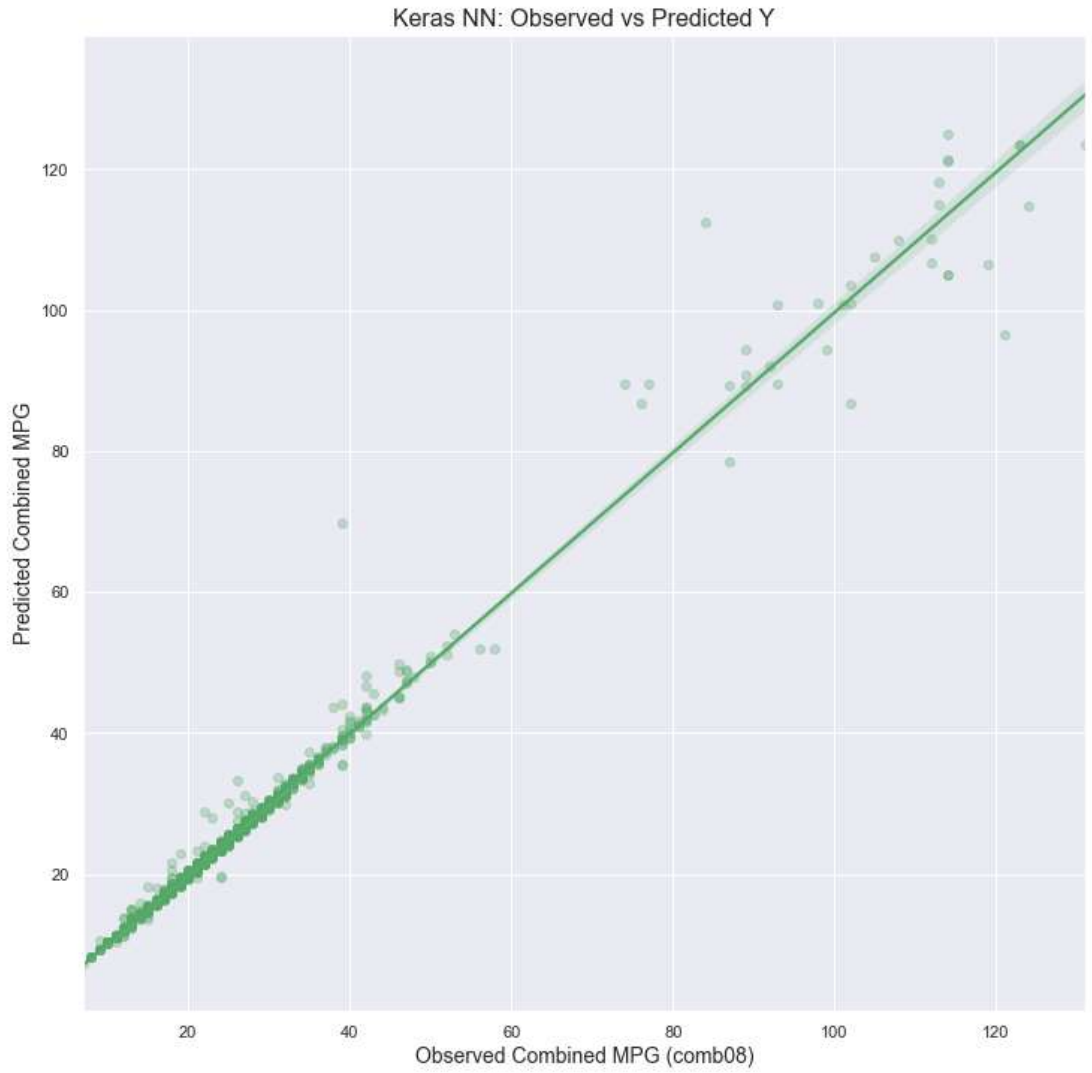
```
nn_plotter = tfdocs.plots.HistoryPlotter(smoothing_std=2)
nn_plotter.plot({'Early Stopping': nn_history}, metric = "mse")
plt.rc("figure", figsize=(14, 10))
plt.rcParams.update({'font.size': 14})
plt.ylabel('MSE [Combined MPG]')
plt.ylim([0, 8])
plt.show()
```



```
y_train_nn_pred = fitted_nn_model(X_train.astype(float))
y_test_nn_pred = fitted_nn_model.predict(X_test.astype(float))
RMSE_nn_train = metrics.mean_squared_error(y_train,\
                                           y_train_nn_pred, squared=False)
RMSE_nn_test = metrics.mean_squared_error(y_test,\
                                           y_test_nn_pred, squared=False)
R2_nn_test = metrics.r2_score(y_test, y_test_nn_pred)
print('RMSE_train: %.4f\tRMSE_test: %.4f\tr2: %.4f' %\
                    (RMSE_nn_train, RMSE_nn_test, R2_nn_test))
```

```
RMSE_train: 0.7012      RMSE_test: 0.7878       r2: 0.9907
```

```
sns.set()
plt.figure(figsize = (12,12))
plt.title('Keras NN: Observed vs Predicted Y', fontsize=16)
plt.ylabel('Predicted Combined MPG', fontsize=14)
sns.regplot(x=y_test, y=y_test_nn_pred, color="g",\
            scatter_kws={'alpha':0.3})
plt.xlabel('Observed Combined MPG (comb08)', fontsize=14)
plt.show()
```



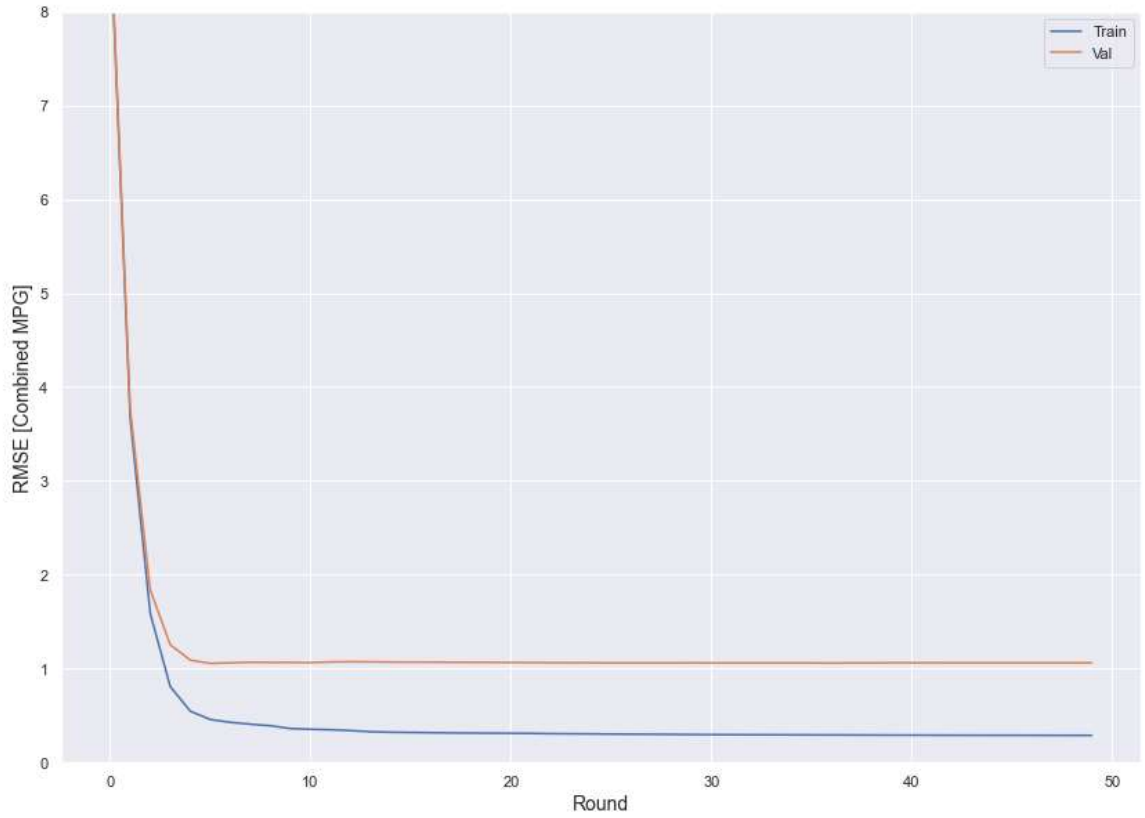
XGBoost

```
dtrain = xgb.DMatrix(X_train, label=y_train)
dval = xgb.DMatrix(X_val, label=y_val)
dtest = xgb.DMatrix(X_test, label=y_test)
```

```
fitted_xgb_model = xgb.XGBRegressor(max_depth=7, learning_rate=0.6,\
                    n_jobs=4, objective='reg:squarederror',\
                    random_state=rand, n_estimators=50).\
                    fit(X_train, y_train,\
                    eval_set=[(X_train, y_train),(X_val, y_val)],\
                    eval_metric='rmse')
```

w hidden output

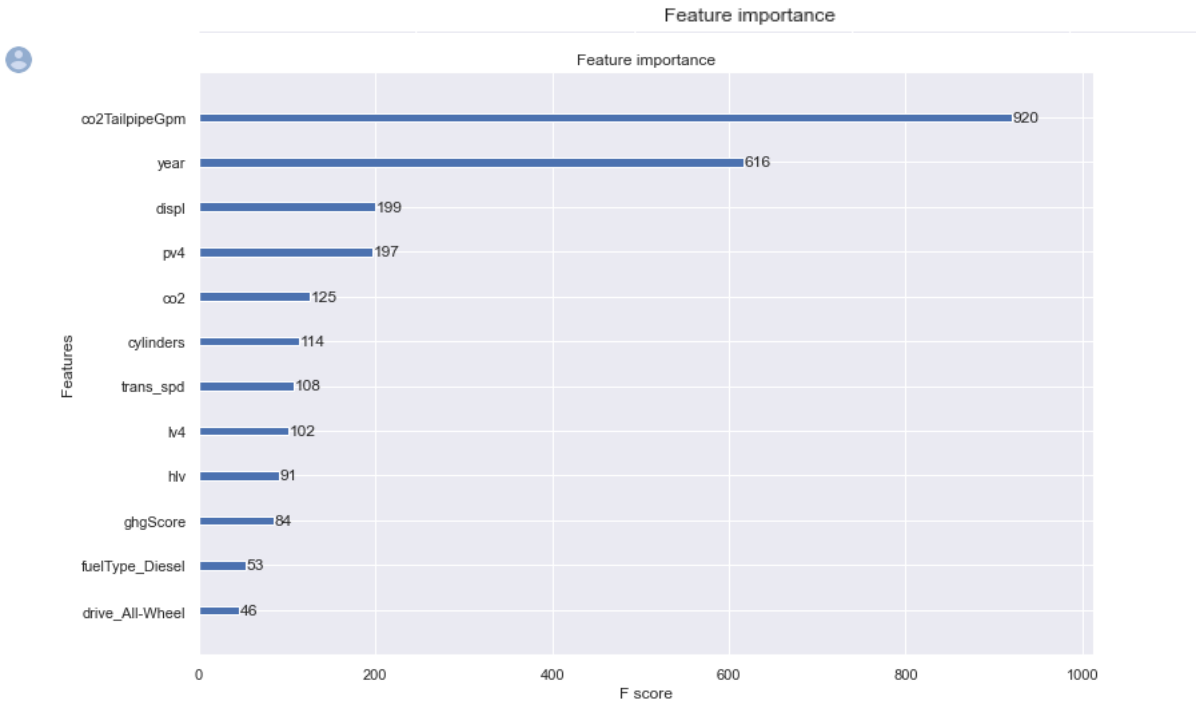
```
plt.figure(figsize=(14, 10))
plt.rcParams.update({'font.size': 14})
plt.plot(fitted_xgb_model.evals_result()['validation_0']['rmse'])
plt.plot(fitted_xgb_model.evals_result()['validation_1']['rmse'])
plt.ylabel('RMSE [Combined MPG]', fontsize=14)
plt.ylim([0, 8])
plt.xlabel('Round', fontsize=14)
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```



```
y_train_xgb_pred = fitted_xgb_model.predict(X_train)
y_test_xgb_pred = fitted_xgb_model.predict(X_test)
RMSE_xgb_train = math.sqrt(\
                    metrics.mean_squared_error(y_train,\
                                               y_train_xgb_pred))
RMSE_xgb_test = math.sqrt(\
                    metrics.mean_squared_error(y_test,
                                               y_test_xgb_pred))
R2_xgb_test = metrics.r2_score(y_test, y_test_xgb_pred)
print('RMSE_train: %.4f\tRMSE_test: %.4f\tr2: %.4f' %\
                    (RMSE_xgb_train, RMSE_xgb_test, R2_xgb_test))
```

```
RMSE_train: 0.2974      RMSE_test: 0.6809       r2: 0.9930
```

```
sns.set()
fig, ax = plt.subplots(figsize=(12, 8))
xgb.plot_importance(fitted_xgb_model, max_num_features=12, ax=ax,\
                    importance_type="weight")
plt.show()
```



SHAP Summary and Dependence Plots

```
shap_xgb_explainer = shap.TreeExplainer(fitted_xgb_model)
```

```
background = X_train.iloc[np.random.choice(X_train.shape[0], 150, replace=False)]
print(background.shape)
shap_nn_explainer = shap.GradientExplainer(fitted_nn_model,\
                                           background.astype(float).values)
```

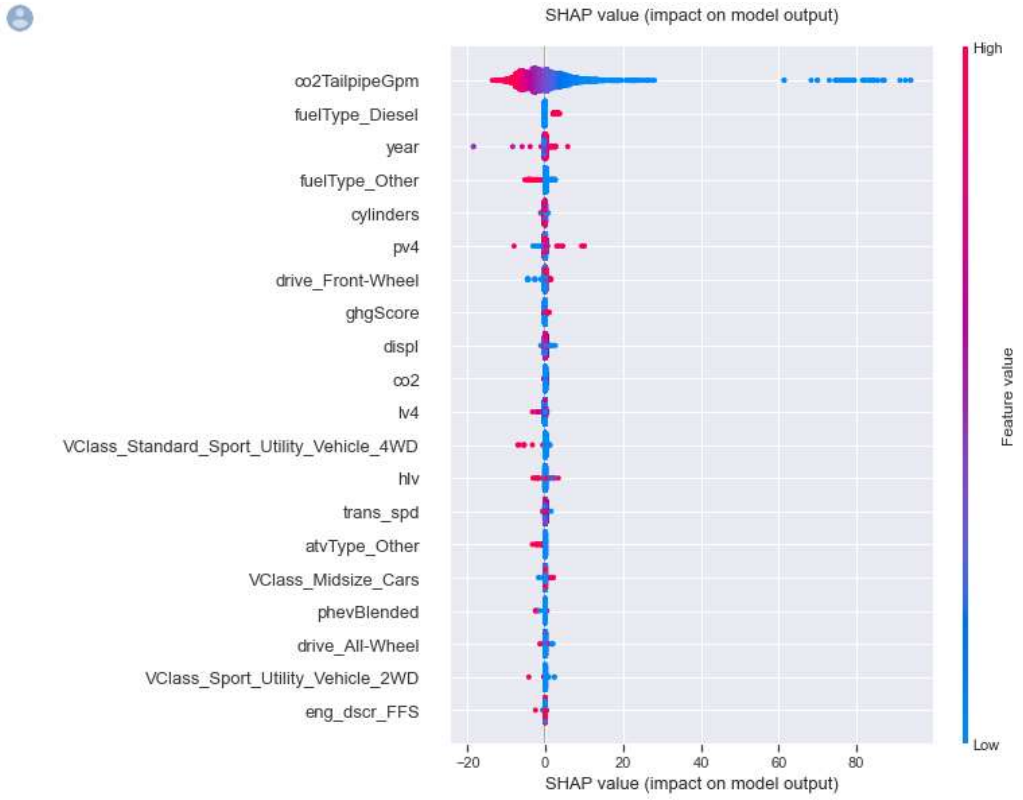
```
(150, 81)
Using plaidml.keras.backend backend.
```

```
shap_xgb_values_train = shap_xgb_explainer.shap_values(X_train)
print(shap_xgb_values_train.shape)
shap_xgb_values_test = shap_xgb_explainer.shap_values(X_test)
print(shap_xgb_values_test.shape)
```

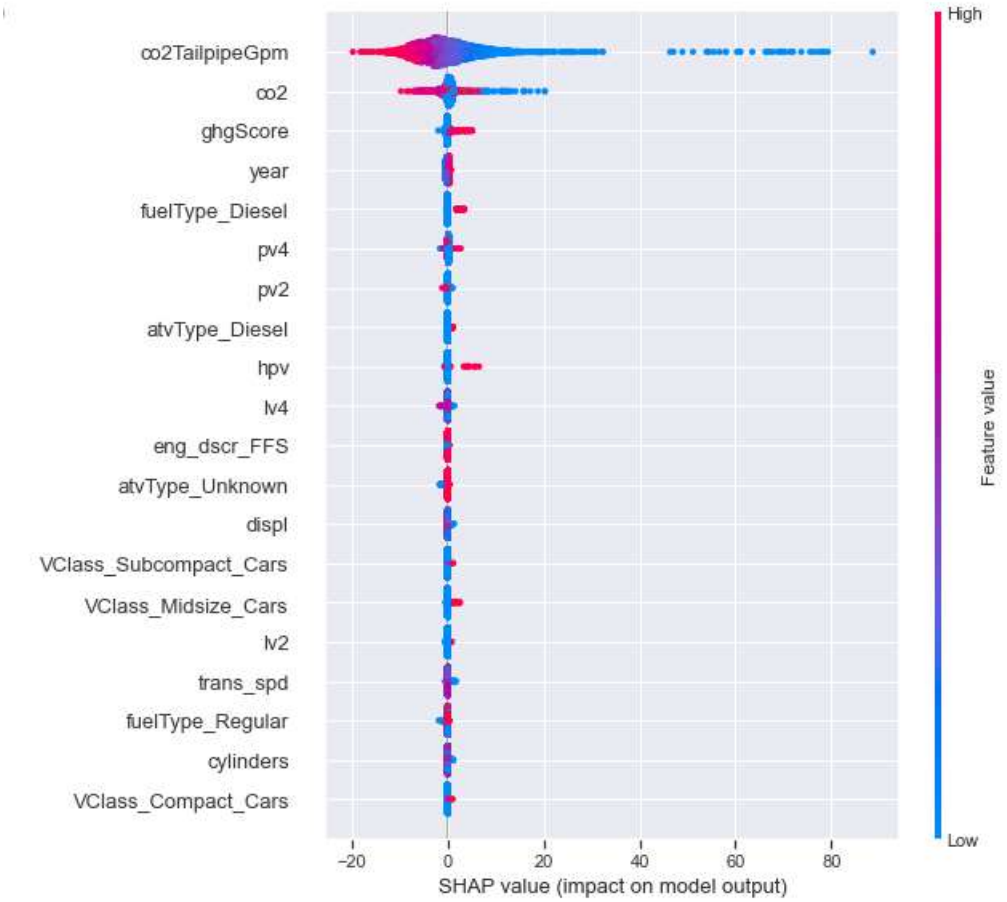
```
(29455, 81)
(6498, 81)
```

```
shap_nn_values_test = shap_nn_explainer.shap_values(X_test.astype(float).values)
print(type(shap_nn_values_test))
print(shap_nn_values_test[0].shape)
```

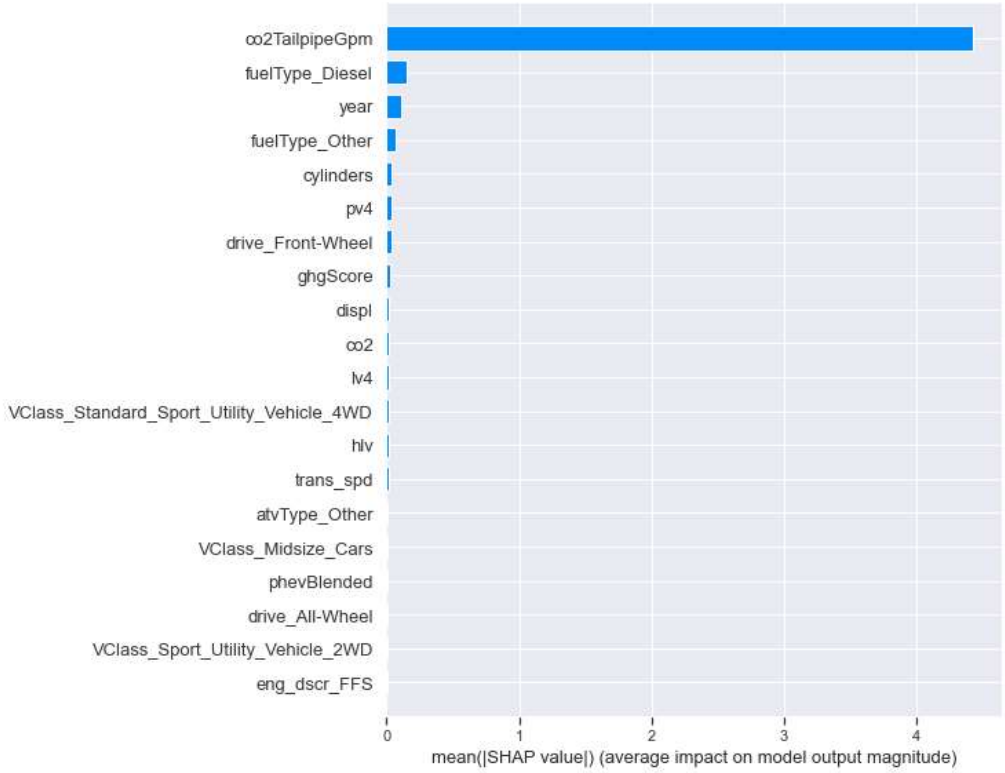
```
<class 'list'>
(6498, 81)
```

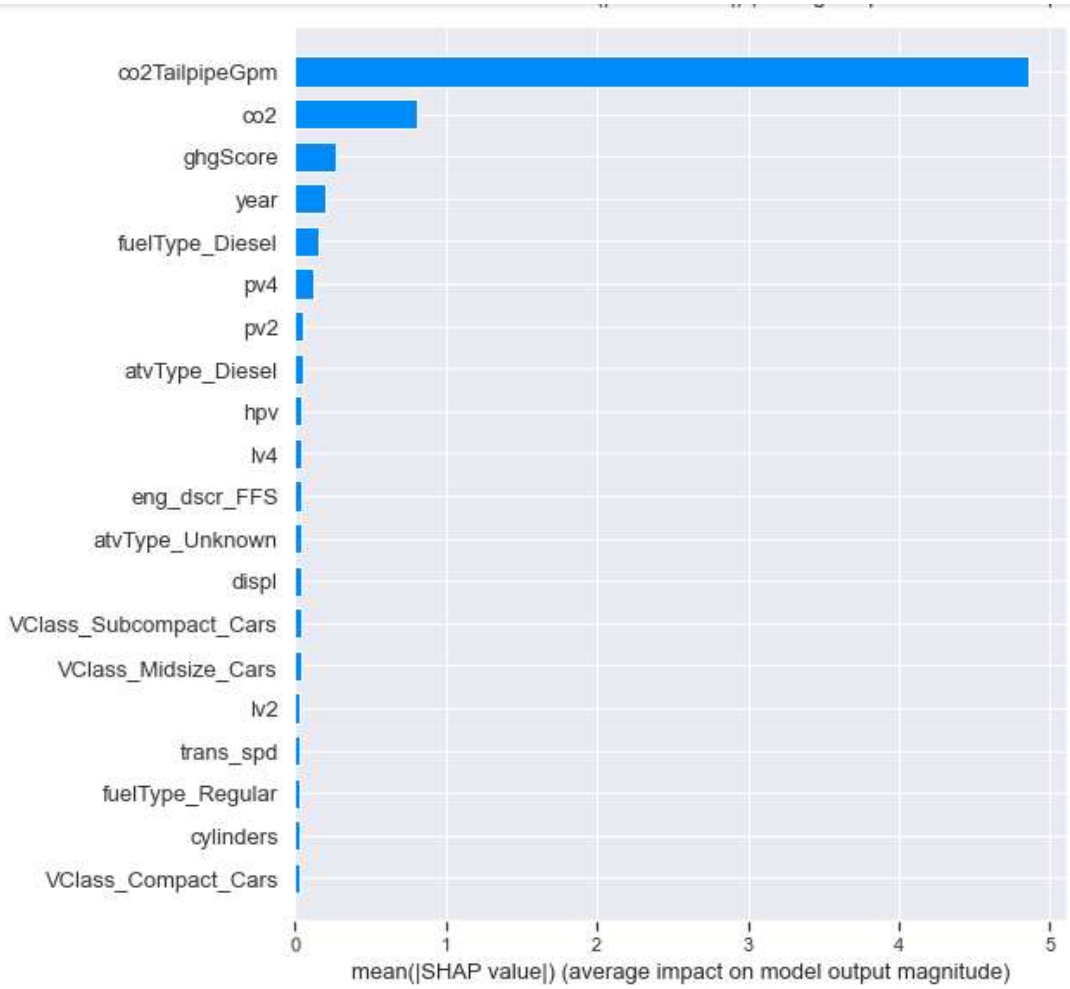

SHAP value (impact on model output)
co2TailpipeGpm
fuelType_Diesel
year
fuelType_Other
cylinders
pv4
drive_Front-Wheel
ghgScore
displ
co2
lv4
VClass_Standard_Sport_Utility_Vehicle_4WD
hlv
trans_spd
atvType_Other
VClass_Midsize_Cars
phevBlended
drive_All-Wheel
VClass_Sport_Utility_Vehicle_2WD
eng_dscr_FFS
−20
0
20
40
60
80
SHAP value (impact on model output)
High
Feature value
Low

```
shap.summary_plot(shap_nn_values_test[0], X_test, plot_type="dot")
```



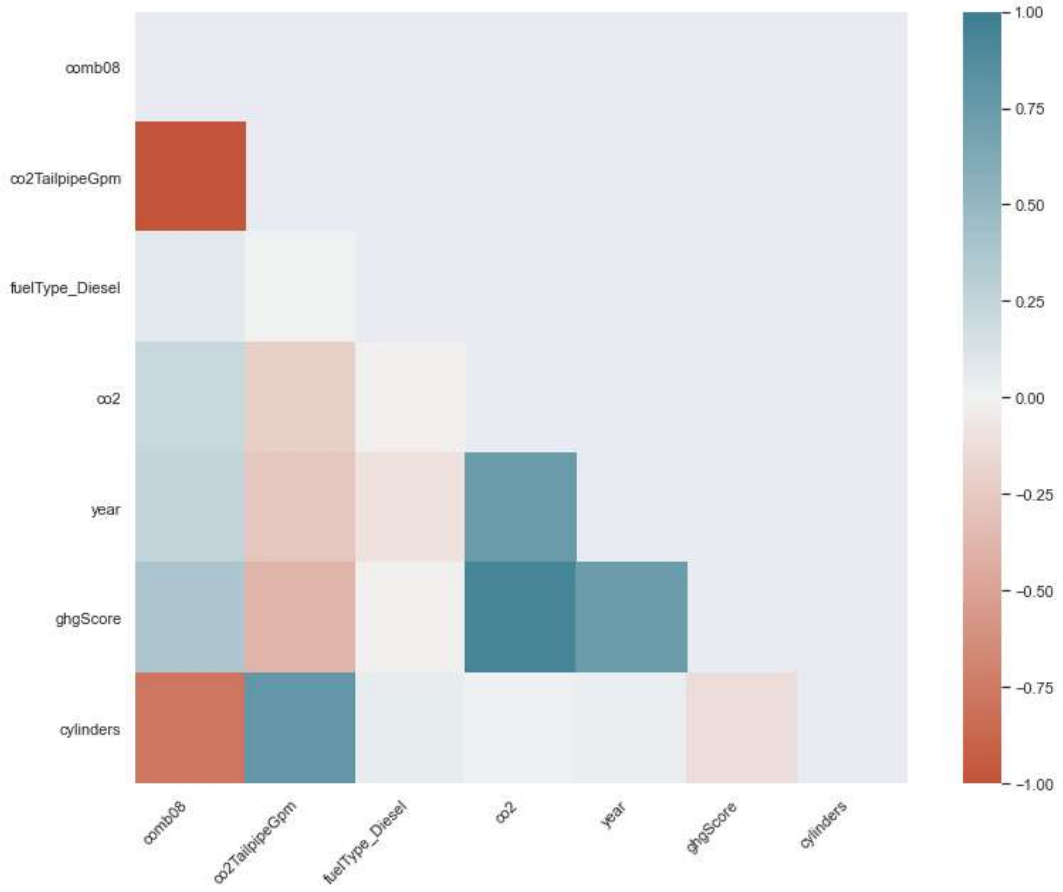
```
shap.summary_plot(shap_xgb_values_test, X_test, plot_type="bar")
shap.summary_plot(shap_nn_values_test[0], X_test, plot_type="bar")
```






```
top_features_l = ['comb08'] + ['co2TailpipeGpm', 'fuelType_Diesel',\
                               'co2', 'year', 'ghgScore', 'cylinders']
top_df = fueleconomy_df.loc[X_train.index, top_features_l]
```

```
corrs = stats.spearmanr(top_df).correlation
mask = np.zeros_like(corrs)
mask[np.triu_indices_from(mask)] = True
ax = sns.heatmap(
    corrs,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 220, n=200),
    xticklabels=top_df.columns,
    yticklabels=top_df.columns,
    mask=mask,
    square=True
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
)
plt.show()
```



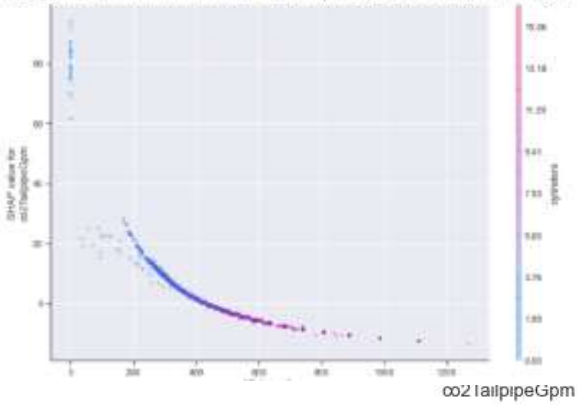

```
print('spearman\tco2TailpipeGpm→comb08\tcorr: %.3f\tp-val: %.4f' %
      (stats.spearmanr(X_train.co2TailpipeGpm.values, top_df.comb08.values)))
print('point-biserial\tfuelType_Diesel→comb08\tcorr: %.3f\tp-val: %.4f' %
      (stats.pointbiserialr(top_df.fuelType_Diesel.values, top_df.comb08.values)))
print('spearman\tco2→comb08\t\tcorr: %.3f\tp-val: %.4f' %
      (stats.spearmanr(X_train.co2.values, top_df.comb08.values)))
print('spearman\tyear→comb08\t\tcorr: %.3f\tp-val: %.4f' %
      (stats.spearmanr(X_train.year.values, top_df.comb08.values)))
print('spearman\tghgScore→comb08\t\tcorr: %.3f\tp-val: %.4f' %
      (stats.spearmanr(top_df.ghgScore.values, top_df.comb08.values)))
print('spearman\tcylinders→comb08\tcorr: %.3f\tp-val: %.4f' %
      (stats.spearmanr(X_train.cylinders.values, top_df.comb08.values)))
```

```
spearman        co2TailpipeGpm→comb08   corr: -0.994    p-val: 0.0000
point-biserial  fuelType_Diesel→comb08  corr: 0.059     p-val: 0.0000
spearman        co2→comb08              corr: 0.220     p-val: 0.0000
spearman        year→comb08             corr: 0.251     p-val: 0.0000
spearman        ghgScore→comb08         corr: 0.371     p-val: 0.0000
spearman        cylinders→comb08        corr: -0.783    p-val: 0.0000
```

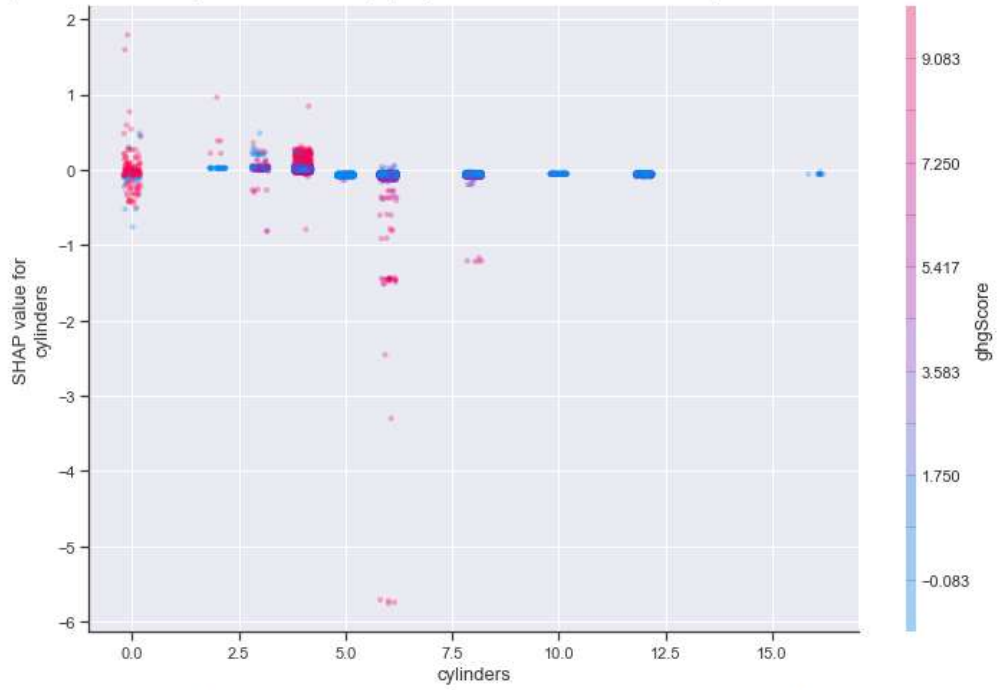
SHAP Dependence Plots

```
shap.dependence_plot("co2TailpipeGpm", shap_xgb_values_test, X_test,\
                     interaction_index="cylinders", show=False, alpha=0.3)
fig = plt.gcf()
fig.set_size_inches(11,8)
plt.show()
print('spearman\tcylinders→co2TailpipeGpm\tcorr: %.3f\tp-val: %.4f' %
      (stats.spearmanr(X_train.cylinders.values, X_train.co2TailpipeGpm.values)))
shap.dependence_plot("cylinders", shap_xgb_values_train, X_train,\
                     interaction_index="ghgScore", show=False,\
                     x_jitter=0.4, alpha=0.3)
fig = plt.gcf()
fig.set_size_inches(11,8)
plt.show()
print('spearman\tghgScore→cylinders\t\tcorr: %.3f\tp-val: %.4f' %
      (stats.spearmanr(top_df.ghgScore.values, top_df.cylinders.values)))
```

```
Passing parameters norm and vmin/vmax simultaneously is deprecated since 3.3 and will become an error two minor releases later. Please pass vmin/vmax directly to the norm when creating it.
```

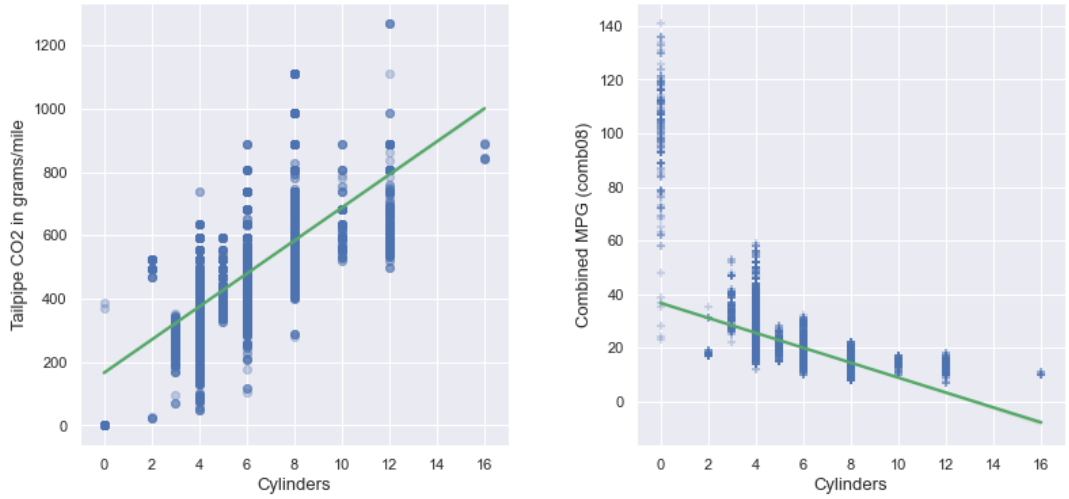



```
spearman        cylinders→co2TailpipeGpm        corr: 0.789     p-val: 0.0000
```

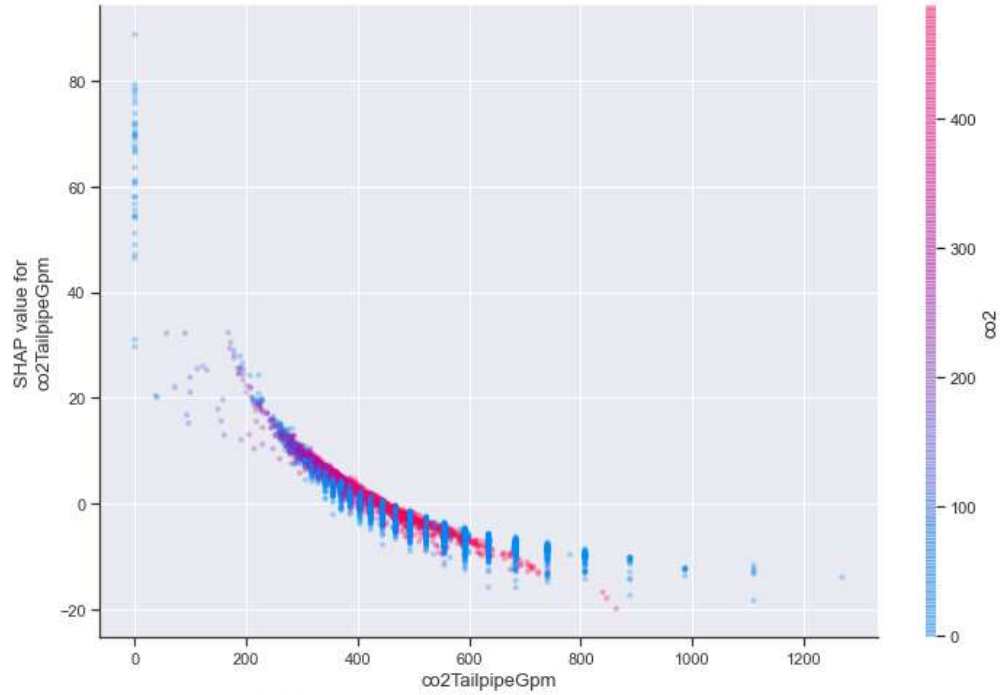



```
spearman        ghgScore→cylinders              corr: -0.123    p-val: 0.0000
```

```
fig, axs = plt.subplots(1, 2, figsize = (13,6))
fig.subplots_adjust(hspace=0, wspace=0.3)
sns.regplot(x=X_train.cylinders, y=X_train.co2TailpipeGpm, ax=axs[0],\
            scatter_kws={'alpha':0.3}, line_kws={'color':'g'})
axs[0].set_ylabel('Tailpipe CO2 in grams/mile', fontsize=13)
axs[0].set_xlabel('Cylinders', fontsize=13)
axs[0].set_xlim([-1,17])
sns.regplot(x=X_train.cylinders, y=y_train, ax=axs[1], marker="+",\
            scatter_kws={'alpha':0.3}, line_kws={'color':'g'})
axs[1].set_ylabel('Combined MPG (comb08)', fontsize=13)
axs[1].set_xlabel('Cylinders', fontsize=13)
axs[1].set_xlim([-1,17])
plt.show()
```

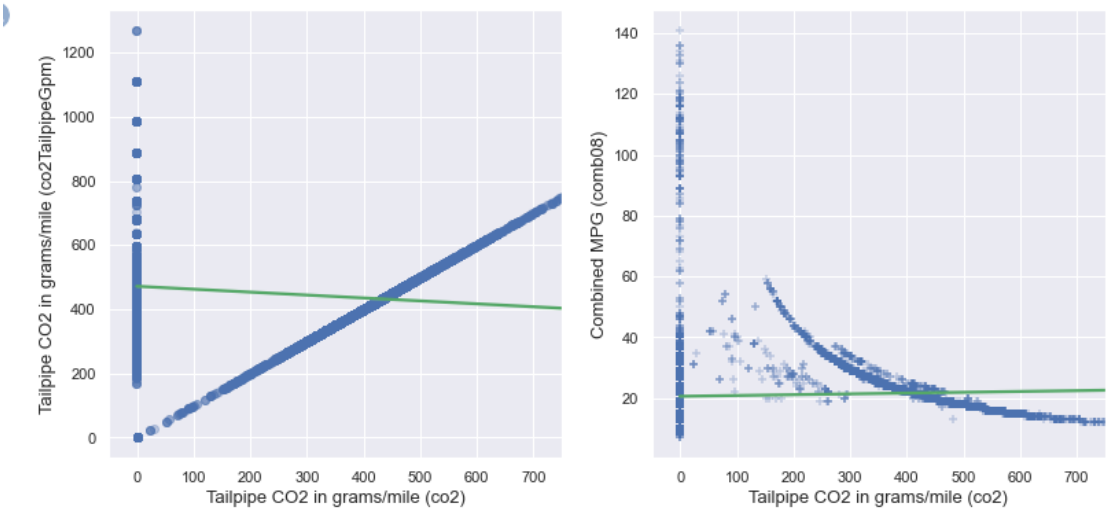


```
shap.dependence_plot("co2TailpipeGpm", shap_nn_values_test[0],\
                     X_test, alpha=0.3,\
                     interaction_index="co2", show=False)
fig = plt.gcf()
fig.set_size_inches(12,8)
plt.show()
print('spearman\tco2→co2TailpipeGpm\tcorr: %.3f\tp-val: %.4f' %
      (stats.spearmanr(X_train.co2.values, X_train.co2TailpipeGpm.values)))
```

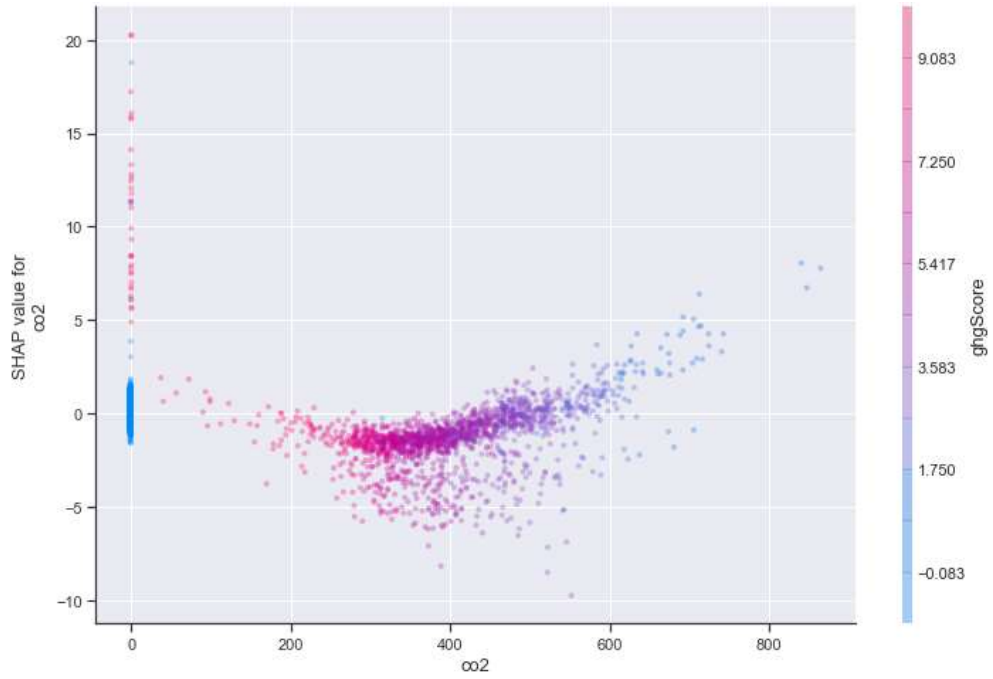



```
spearman        co2→co2TailpipeGpm      corr: -0.224    p-val: 0.0000
```

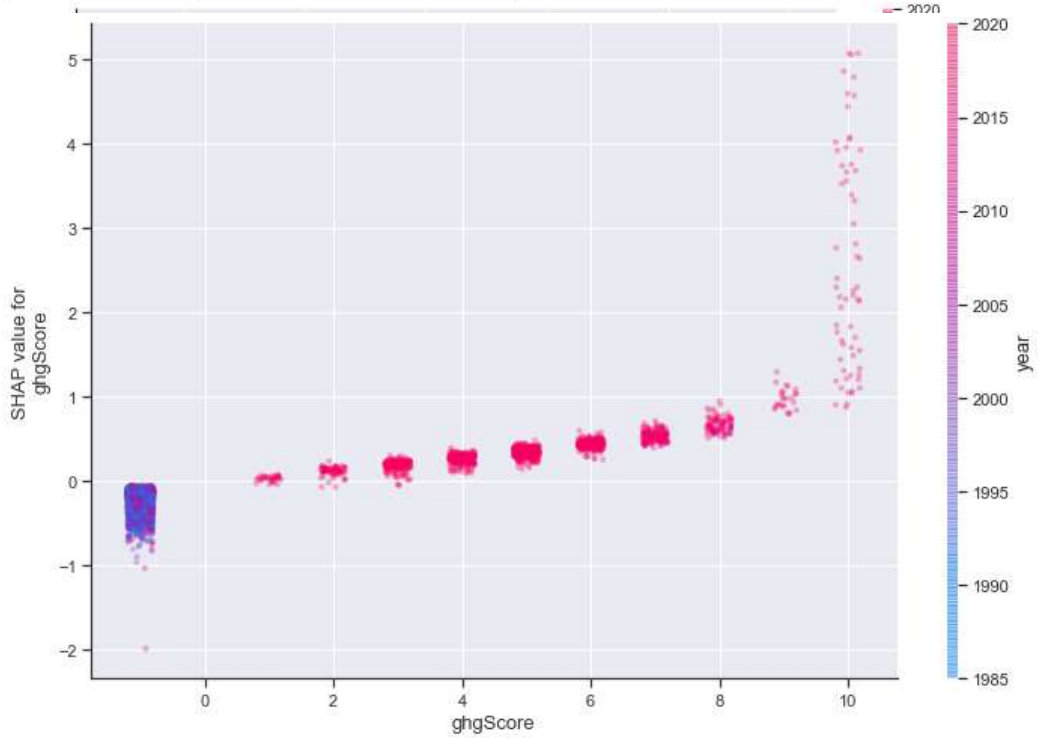
```
fig, axs = plt.subplots(1, 2, figsize = (13,6))
sns.regplot(x=X_train.co2, y=X_train.co2TailpipeGpm, ax=axs[0],\
            scatter_kws={'alpha':0.3}, line_kws={'color':'g'})
axs[0].set_ylabel('Tailpipe CO2 in grams/mile (co2TailpipeGpm)',\
                  fontsize=13)
axs[0].set_xlabel('Tailpipe CO2 in grams/mile (co2)', fontsize=13)
axs[0].set_xlim([-50, 750])
sns.regplot(x=X_train.co2, y=y_train, ax=axs[1], marker="+",\
            scatter_kws={'alpha':0.3}, line_kws={'color':'g'})
axs[1].set_ylabel('Combined MPG (comb08)', fontsize=13)
axs[1].set_xlabel('Tailpipe CO2 in grams/mile (co2)', fontsize=13)
axs[1].set_xlim([-50, 750])
plt.show()
```




```
shap.dependence_plot("co2", shap_nn_values_test[0],\
                     X_test, alpha=0.3, x_jitter=10,\
                     interaction_index="ghgScore", show=False)
fig = plt.gcf()
fig.set_size_inches(12,8)
plt.show()
print('spearman\tghgScore→co2\tcorr: %.3f\tp-val: %.4f' %
      (stats.spearmanr(top_df.ghgScore.values, top_df.co2.values)))
shap.dependence_plot("ghgScore", shap_nn_values_test[0],\
                     X_test, alpha=0.3, x_jitter=0.4,\
                     interaction_index="year", show=False)
fig = plt.gcf()
fig.set_size_inches(12,8)
plt.show()
print('spearman\tghgScore→year\tcorr: %.3f\tp-val: %.4f' %
      (stats.spearmanr(top_df.ghgScore.values, top_df.year.values)))
```

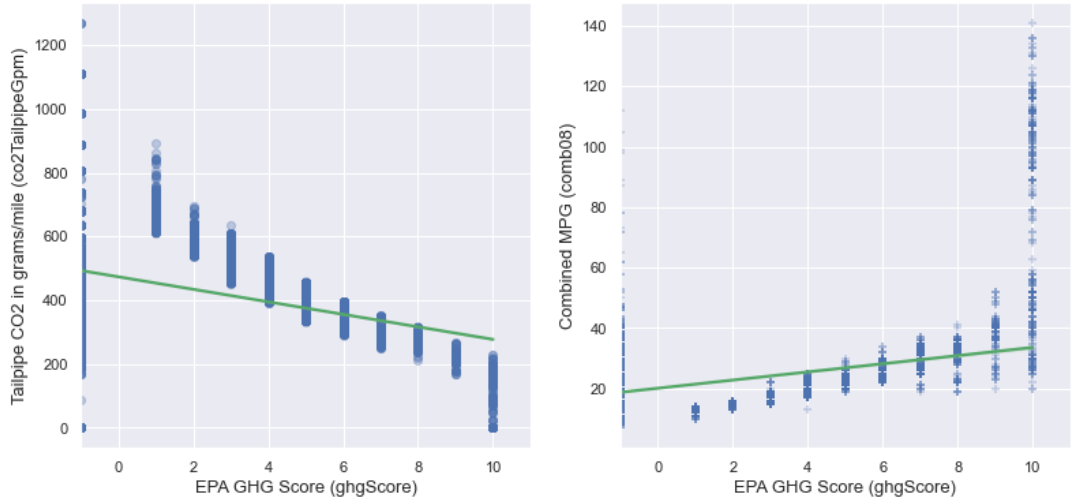


spearman ghgScore→co2 corr: 0.941 p-val: 0.0000



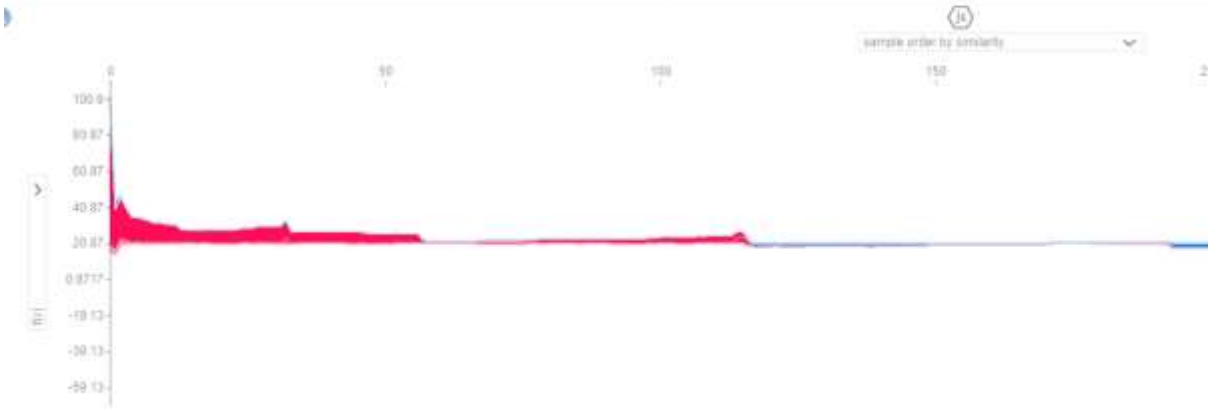
spearman ghgScore→year corr: 0.745 p-val: 0.0000

```
fig, axs = plt.subplots(1, 2, figsize = (13,6))
sns.regplot(x=X_train.ghgScore, y=X_train.co2TailpipeGpm, ax=axs[0],\
            scatter_kws={'alpha':0.3}, line_kws={'color':'g'})
axs[0].set_ylabel('Tailpipe CO2 in grams/mile (co2TailpipeGpm)',\
                  fontsize=13)
axs[0].set_xlabel('EPA GHG Score (ghgScore)', fontsize=13)
axs[0].set_xlim([-1, 11])
sns.regplot(x=X_train.ghgScore, y=y_train, ax=axs[1], marker="+",\
            scatter_kws={'alpha':0.3}, line_kws={'color':'g'})
axs[1].set_ylabel('Combined MPG (comb08)', fontsize=13)
axs[1].set_xlabel('EPA GHG Score (ghgScore)', fontsize=13)
axs[1].set_xlim([-1, 11])
plt.show()
```



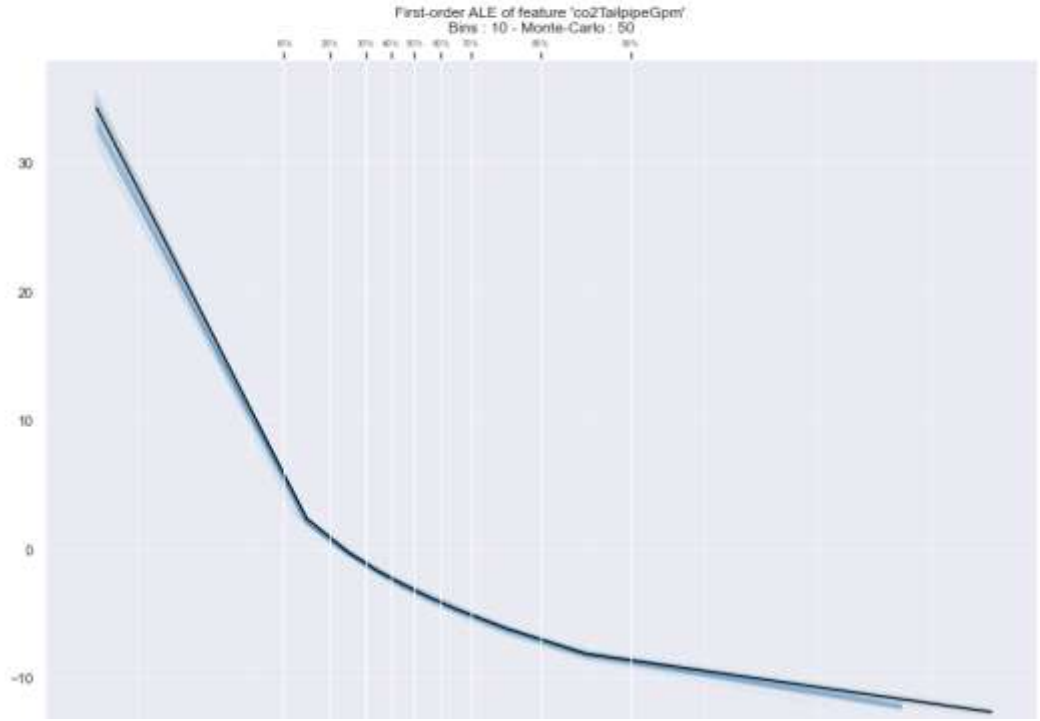
SHAP Force Plots

```
sample_test_size = 0.05
sample_test_idx = np.random.choice(X_test.shape[0],
                                   math.ceil(X_test.shape[0]*sample_test_size),\
                                   replace=False)
shap.initjs()
shap.force_plot(shap_xgb_explainer.expected_value,\
                shap_xgb_values_test[sample_test_idx],\
                X_test.iloc[sample_test_idx])
```

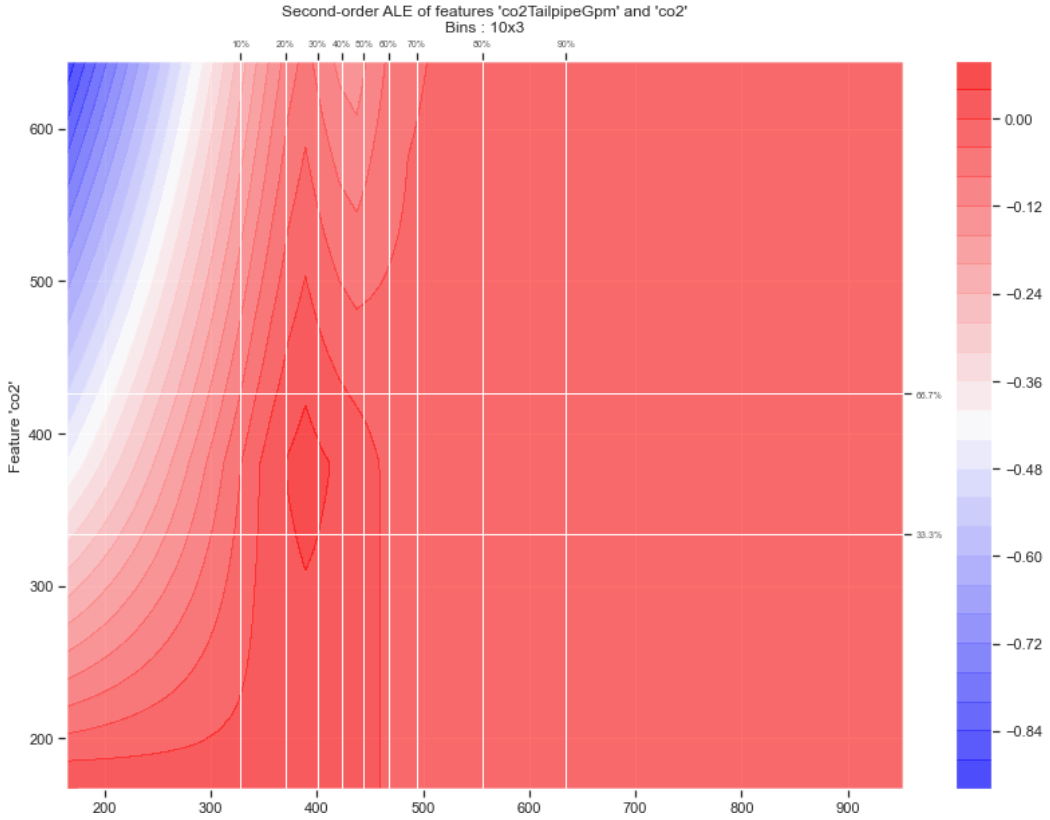


ALE Plots

```
for feature_name in ['co2TailpipeGpm', 'co2', 'ghgScore',\
                     'year', 'cylinders']:
    plt.rc("figure", figsize=(14, 10))
    ale_plot(
        fitted_xgb_model, X_test, [feature_name], bins=10,\
        monte_carlo=True, monte_carlo_rep=50,\
        monte_carlo_ratio=0.4
    )
    plt.show()
```



```
for interaction in [['co2TailpipeGpm', 'co2'], ['co2TailpipeGpm', 'ghgScore'],\
                    ['cylinders', 'co2TailpipeGpm'], ['year', 'co2TailpipeGpm']]:
    plt.rc("figure", figsize=(14, 10))
    ale_plot(
        fitted_xgb_model, X_test, interaction, bins=[10,10]
    )
    plt.show()
```




```
fitted_dt_surrogate = tree.DecisionTreeRegressor(max_depth=7,\
                                                 random_state=rand).\
                                            fit(X_train, y_train_nn_pred)
y_train_dt_pred = fitted_dt_surrogate.predict(X_train)
y_test_dt_pred = fitted_dt_surrogate.predict(X_test)
```

```
fitted_rf_surrogate = RuleFit(max_rules=150, rfmode='regress',\
                              random_state=rand, tree_size=8).\
                      fit(X_train.astype(float).values,\
                          np.array(y_train_nn_pred).squeeze(),\
                          X_train.columns)
y_train_rf_pred = fitted_rf_surrogate.predict(X_train.astype(float).values)
y_test_rf_pred = fitted_rf_surrogate.predict(X_test.astype(float).values)
```

```
#Measure how well Decision Tree replicates Neural Network's predictions
RMSE_dt_nn_train = metrics.mean_squared_error(y_train_nn_pred,\
                                              y_train_dt_pred,\
                                              squared=False)
RMSE_dt_nn_test = metrics.mean_squared_error(y_test_nn_pred,\
                                             y_test_dt_pred,\
                                             squared=False)
R2_dt_nn_test = metrics.r2_score(y_test_nn_pred, y_test_dt_pred)
#Print all metrics
print('RMSE_train: %.4f\tRMSE_test: %.4f\tr2: %.4f' %\
                (RMSE_dt_nn_train, RMSE_dt_nn_test, R2_dt_nn_test))
```

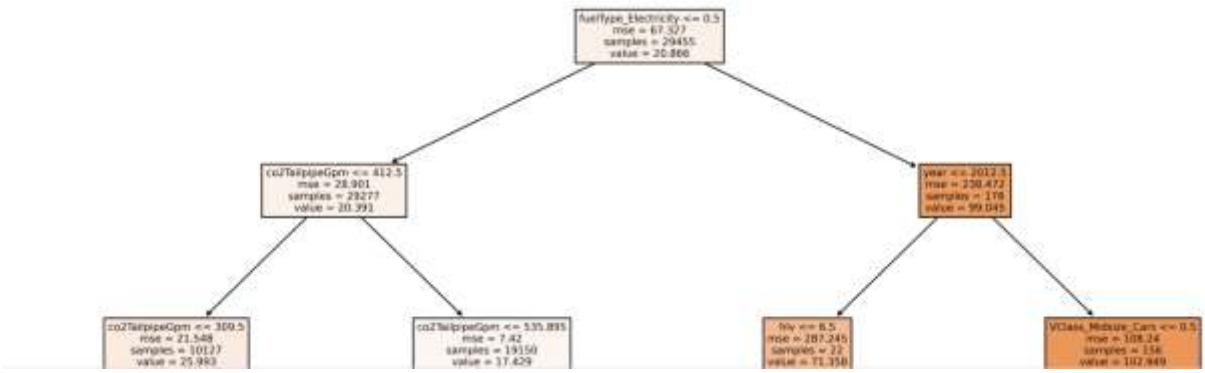
```
RMSE_train: 0.4805      RMSE_test: 0.6100       r2: 0.9944
```

```
#Measure how well Rule Fit replicates Neural Network's predictions
RMSE_rf_nn_train = metrics.mean_squared_error(y_train_nn_pred,\
                                              y_train_rf_pred,\
                                              squared=False)
RMSE_rf_nn_test = metrics.mean_squared_error(y_test_nn_pred,\
                                             y_test_rf_pred,\
                                             squared=False)
R2_rf_nn_test = metrics.r2_score(y_test_nn_pred, y_test_rf_pred)
#Print all metrics
print('RMSE_train: %.4f\tRMSE_test: %.4f\tr2: %.4f' %\
                (RMSE_rf_nn_train, RMSE_rf_nn_test, R2_rf_nn_test))
```

```
RMSE_train: 0.9150      RMSE_test: 0.9049       r2: 0.9877
```

```
fig, axes = plt.subplots(nrows = 1, ncols = 1, figsize = (10,8), dpi=600)
tree.plot_tree(fitted_dt_surrogate,\
               feature_names=X_train.columns.values.tolist(), filled = True, max_depth=2)
fig.show()
```

```
Matplotlib is currently using module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot show the figure.
```



```
text_tree = tree.export_text(fitted_dt_surrogate,\
               feature_names=X_train.columns.values.tolist())
print(text_tree)
```

```
|--- fuelType_Electricity <= 0.50
|   |--- co2TailpipeGpm <= 412.50
|   |   |--- co2TailpipeGpm <= 309.50
|   |   |   |--- co2TailpipeGpm <= 234.43
|   |   |   |   |--- drive_Front-Wheel <= 0.50
|   |   |   |   |   |--- atvType_Other <= 0.50
|   |   |   |   |   |   |--- co2TailpipeGpm <= 197.88
|   |   |   |   |   |   |   |--- value: [49.54]
|   |   |   |   |   |   |--- co2TailpipeGpm >  197.88
|   |   |   |   |   |   |   |--- value: [40.34]
|   |   |   |   |   |--- atvType_Other >  0.50
|   |   |   |   |   |   |--- co2TailpipeGpm <= 165.50
|   |   |   |   |   |   |   |--- value: [32.85]
|   |   |   |   |   |   |--- co2TailpipeGpm >  165.50
|   |   |   |   |   |   |   |--- value: [29.28]
|   |   |   |   |--- drive_Front-Wheel >  0.50
|   |   |   |   |   |--- co2TailpipeGpm <= 194.50
|   |   |   |   |   |   |--- hlv <= 19.50
|   |   |   |   |   |   |   |--- value: [46.85]
|   |   |   |   |   |   |--- hlv >  19.50
|   |   |   |   |   |   |   |--- value: [53.64]
|   |   |   |   |   |--- co2TailpipeGpm >  194.50
|   |   |   |   |   |   |--- co2TailpipeGpm <= 216.88
|   |   |   |   |   |   |   |--- value: [42.81]
|   |   |   |   |   |   |--- co2TailpipeGpm >  216.88
|   |   |   |   |   |   |   |--- value: [40.00]
|   |   |   |--- co2TailpipeGpm >  234.43
|   |   |   |   |--- co2TailpipeGpm <= 284.50
|   |   |   |   |   |--- atvType_Other <= 0.50
|   |   |   |   |   |   |--- co2TailpipeGpm <= 267.95
|   |   |   |   |   |   |   |--- value: [35.23]
|   |   |   |   |   |   |--- co2TailpipeGpm >  267.95
|   |   |   |   |   |   |   |--- value: [32.42]
|   |   |   |   |   |--- atvType_Other >  0.50
|   |   |   |   |   |   |--- startStop_Y <= 0.50
|   |   |   |   |   |   |   |--- value: [29.00]
|   |   |   |   |   |   |--- startStop_Y >  0.50
|   |   |   |   |   |   |   |--- value: [24.16]
```

```
rulefit_df = fitted_rf_surrogate.get_rules()
rulefit_df = rulefit_df[rulefit_df.coef != 0]
rulefit_df.sort_values(by="importance", ascending=False)
```

	rule	type	coef	support	importance
4	co2TailpipeGpm	linear	-0.032542	1.000000	3.663626
132	atvType_EV > 0.5	rule	24.897840	0.008857	2.332828
88	displ > 0.30000001192092896	rule	-14.924096	0.992914	1.251820
167	cylinders <= 1.0 & ghgScore <= 4.5	rule	-17.959679	0.001771	0.755234
14	fuelType_Diesel	linear	3.624268	1.000000	0.596587
...	...	...	...	...	...
38	VClass_Small_Station_Wagons	linear	0.034334	1.000000	0.006595
64	eng_dscr_CA_model	linear	0.013023	1.000000	0.006403
26	drive_Rear-Wheel	linear	0.009481	1.000000	0.004467
177	co2TailpipeGpm <= 86.0 & co2TailpipeGpm <= 377...	rule	0.015664	0.003543	0.000931
92	co2TailpipeGpm <= 535.8947448730469 & co2Tailp...	rule	-0.000027	0.325066	0.000013

89 rows × 5 columns

Chapter 6 (Choco Rating):

```
!pip install --upgrade pandas numpy scikit-learn tensorflow matplotlib seaborn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages (1.3.5)
Collecting pandas
  Downloading pandas-1.5.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.2/12.2 MB 31.5 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (1.21.6)
Collecting numpy
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 31.9 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.8/dist-packages (1.0.2)
Collecting scikit-learn
  Downloading scikit_learn-1.2.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (9.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.8/9.8 MB 19.9 MB/s eta 0:00:00
Requirement already satisfied: tensorflow in /usr/local/lib/python3.8/dist-packages (2.9.2)
Collecting tensorflow
  Downloading tensorflow-2.11.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (588.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 588.3/588.3 MB 1.2 MB/s eta 0:00:00
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (3.2.2)
Collecting matplotlib
  Downloading matplotlib-3.6.3-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (9.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.4/9.4 MB 62.2 MB/s eta 0:00:00
Requirement already satisfied: seaborn in /usr/local/lib/python3.8/dist-packages (0.11.2)
Collecting seaborn
  Downloading seaborn-0.12.2-py3-none-any.whl (293 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 293.3/293.3 KB 26.9 MB/s eta 0:00:00
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (3.1.0)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.7.3)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (4.4.0)
```

```
import math
import machine_learning_datasets as mldatasets
import pandas as pd
import numpy as np
import re
import nltk
from nltk.probability import FreqDist
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn import metrics, svm
from sklearn.feature_extraction.text import TfidfVectorizer
import lightgbm as lgb
import matplotlib.pyplot as plt
import seaborn as sns
import shap
import lime
import lime.lime_tabular
from lime.lime_text import LimeTextExplainer
```

```
nltk.download('stopwords')
nltk.download('punkt')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

```
[nltk_data] Downloading package stopwords to /Users/serg/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /Users/serg/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
chocolateratings_df = mldatasets.load("chocolate-bar-ratings")
```

```
https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python/raw/master/datasets/chocolate-ratings.zip downloaded to /Users/serg/EXTEND/PROJECTS/InterpretableMLBook/programm
/Users/serg/EXTEND/PROJECTS/InterpretableMLBook/programming/Chapter6/data/chocolate-ratings.zip uncompressed to /Users/serg/EXTEND/PROJECTS/InterpretableMLBook/programming/Chapter6/data/cho
1 dataset files found in /Users/serg/EXTEND/PROJECTS/InterpretableMLBook/programming/Chapter6/data/chocolate-ratings folder
parsing /Users/serg/EXTEND/PROJECTS/InterpretableMLBook/programming/Chapter6/data/chocolate-ratings/chocolate.csv
```

```
chocolateratings_df
```

	company	company_location	review_date	country_of_bean_origin	cocoa_percent	rating	counts_of_ingredients	beans	cocoa_butter	vanilla	lecithin	salt	
0	5150	U.S.A.	2019	Madagascar	76.0	3.75	3	have_bean	have_cocoa_butter	have_not_vanila	have_not_lecithin	have_not_salt	
1	5150	U.S.A.	2019	Dominican republic	76.0	3.50	3	have_bean	have_cocoa_butter	have_not_vanila	have_not_lecithin	have_not_salt	
2	5150	U.S.A.	2019	Tanzania	76.0	3.25	3	have_bean	have_cocoa_butter	have_not_vanila	have_not_lecithin	have_not_salt	
3	A. Morin	France	2012	Peru	63.0	3.75	4	have_bean	have_cocoa_butter	have_not_vanila	have_lecithin	have_not_salt	
4	A. Morin	France	2012	Bolivia	70.0	3.50	4	have_bean	have_cocoa_butter	have_not_vanila	have_lecithin	have_not_salt	
...	...	...	...	...	...	...	...	...	...	...	...	...	
2219	Zotter	Austria	2014	Blend	60.0	2.75	4	have_bean	have_cocoa_butter	have_not_vanila	have_not_lecithin	have_salt	ha
2220	Zotter	Austria	2017	Colombia	75.0	3.75	3	have_bean	have_cocoa_butter	have_not_vanila	have_not_lecithin	have_not_salt	
2221	Zotter	Austria	2018	Belize	72.0	3.50	3	have_bean	have_cocoa_butter	have_not_vanila	have_not_lecithin	have_not_salt	
2222	Zotter	Austria	2018	Congo	70.0	3.25	3	have_bean	have_cocoa_butter	have_not_vanila	have_not_lecithin	have_not_salt	
2223	Zotter	Austria	2018	Blend	75.0	3.00	3	have_bean	have_cocoa_butter	have_not_vanila	have_not_lecithin	have_not_salt	

2224 rows × 10 columns

```
tastes_df = chocolateratings_df[['first_taste', 'second_taste', 'third_taste', 'fourth_taste']]
chocolateratings_df = chocolateratings_df.\
                    drop(['first_taste', 'second_taste', 'third_taste', 'fourth_taste'], axis=1)
tastes_df.head(90).tail(10)
```

	first_taste	second_taste	third_taste	fourth_taste
80	oily	vegetal	nutty	cocoa
81	oily	vanilla	melon	cocoa
82	rich	sour	mild smoke	NaN
83	fruity	sour	NaN	NaN
84	high roast	high astringnet	NaN	NaN
85	smokey	savory	NaN	NaN
86	sandy	roasty	nutty	NaN
87	roasty	brownie	nutty	NaN
88	red wine	rich cocoa	long	NaN
89	creamy	fruit	cocoa	NaN

```
chocolateratings_df = mldatasets.make_dummies_with_limits(chocolateratings_df,\
                                        'company_location', 0.03333)
chocolateratings_df = mldatasets.make_dummies_with_limits(chocolateratings_df,\
                                        'country_of_bean_origin', 0.03333)
```

```
chocolateratings_df = chocolateratings_df.\
                                drop(['beans'], axis=1)
binary_features = ['cocoa_butter', 'vanilla', 'lecithin', 'salt',\
                   'sugar', 'sweetener_without_sugar']
chocolateratings_df[binary_features] =\
    chocolateratings_df[binary_features].\
        apply(lambda x: np.where(x.str.contains('_not_'), 0, 1))
```

```
stop = stopwords.words('english')
```

```
trans_dict = {'?':'', '&':'', 'overly intense':'intensest',\
              'overly sweet':'sweetest', 'overly tart':'tartest',\
              'sl. bitter':'bitterness', 'sl. burnt':'burntness',\
              'sl. sweet':'sweetness', 'sl. dry':'dryness',\
              'sl. chalky':'chalkiness', 'sl. Burnt':'burntness',\
              'hints fruit':'fruitiness', 'hint fruit':'fruitiness',\
              'high acid':'acidic', 'high acidity':'acidic',\
              'moderate acidity':'acid', 'high roast':'roast',\
              'astringcy':'astringent', 'astringnet':'astringent',\
              'full body':'robust', 'astringency':'astringent',\
              'high astringent':'acidic', 'rich cocoa':'rich',\
              'mild bitter':'bitterish', 'fruit long':'fruit',\
              'base cocoa':'basic', 'basic cocoa':'basic', '-like':'',\
              'smomkey':'smokey', 'true':'real', '(n)':'', '/':' ',\
              '-':' ', ' +':' ' }
trans_regex = re.compile("|".join(map(re.escape, trans_dict.keys( ))))
```

```
chocolateratings_df['tastes'] = tastes_s
chocolateratings_df = mldatasets.make_dummies_from_dict(chocolateratings_df,\
                                                        'tastes', commontastes_l)
```

```
chocolateratings_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2224 entries, 0 to 2223
Data columns (total 46 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   company                                     2224 non-null   object
 1   review_date                                 2224 non-null   int64
 2   cocoa_percent                               2224 non-null   float64
 3   rating                                      2224 non-null   float64
 4   counts_of_ingredients                       2224 non-null   int64
 5   cocoa_butter                                2224 non-null   int64
 6   vanilla                                     2224 non-null   int64
 7   lecithin                                    2224 non-null   int64
 8   salt                                        2224 non-null   int64
 9   sugar                                       2224 non-null   int64
 10  sweetener_without_sugar                     2224 non-null   int64
 11  company_location_Canada                     2224 non-null   uint8
 12  company_location_France                     2224 non-null   uint8
 13  company_location_Other                      2224 non-null   uint8
 14  company_location_U.S.A                      2224 non-null   uint8
 15  company_location_U.k.                       2224 non-null   uint8
 16  country_of_bean_origin_Blend                2224 non-null   uint8
 17  country_of_bean_origin_Dominican_republic   2224 non-null   uint8
 18  country_of_bean_origin_Ecuador              2224 non-null   uint8
 19  country_of_bean_origin_Madagascar           2224 non-null   uint8
 20  country_of_bean_origin_Nicaragua            2224 non-null   uint8
 21  country_of_bean_origin_Other                2224 non-null   uint8
 22  country_of_bean_origin_Peru                 2224 non-null   uint8
 23  country_of_bean_origin_Venezuela            2224 non-null   uint8
 24  tastes cocoa                                2224 non-null   int64
```

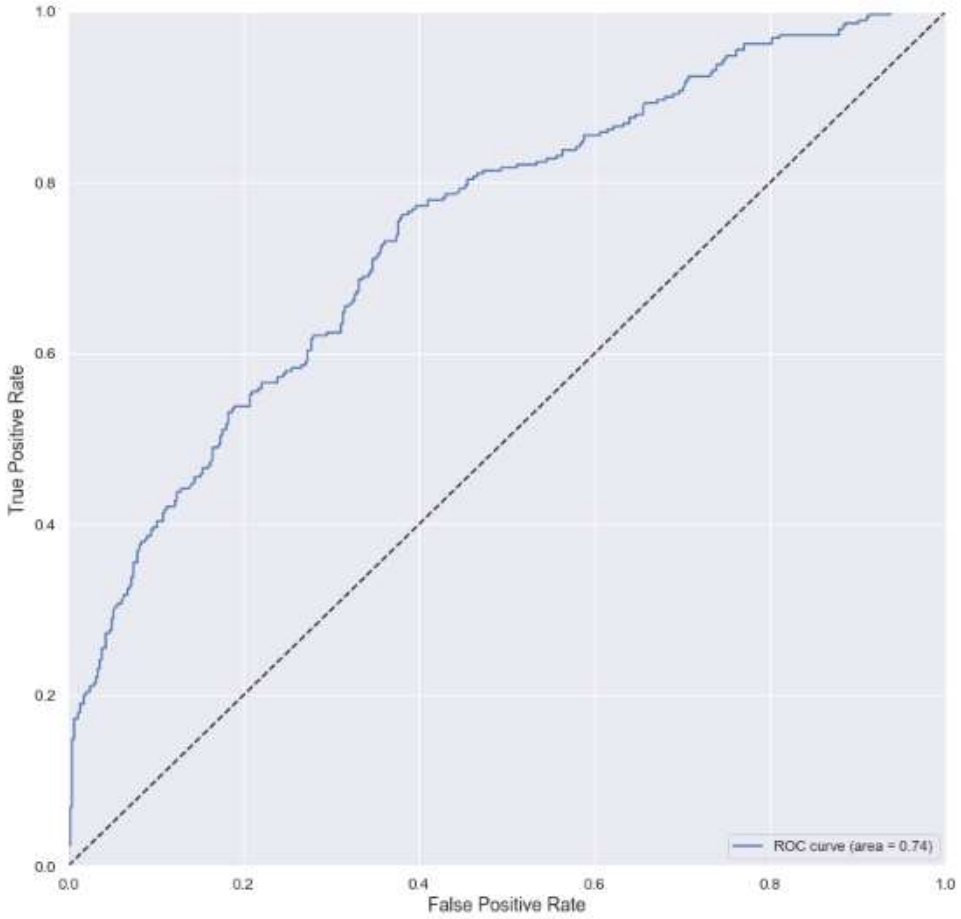
```
 25  tastes_rich                                2224 non-null   int64
 26  tastes_fatty                               2224 non-null   int64
 27  tastes_roasty                              2224 non-null   int64
 28  tastes_nutty                               2224 non-null   int64
 29  tastes_sweet                               2224 non-null   int64
 30  tastes_sandy                               2224 non-null   int64
 31  tastes_sour                                2224 non-null   int64
 32  tastes_intense                             2224 non-null   int64
 33  tastes_mild                                2224 non-null   int64
 34  tastes_fruit                               2224 non-null   int64
 35  tastes_sticky                              2224 non-null   int64
 36  tastes_earthy                              2224 non-null   int64
 37  tastes_spice                               2224 non-null   int64
 38  tastes_molasses                            2224 non-null   int64
 39  tastes_floral                              2224 non-null   int64
 40  tastes_spicy                               2224 non-null   int64
 41  tastes_woody                               2224 non-null   int64
 42  tastes_coffee                              2224 non-null   int64
 43  tastes_berry                               2224 non-null   int64
 44  tastes_vanilla                             2224 non-null   int64
 45  tastes_creamy                              2224 non-null   int64
dtypes: float64(2), int64(30), object(1), uint8(13)
memory usage: 601.7+ KB
```

```
rand = 9
y = chocolateratings_df['rating'].\
                            apply(lambda x: 1 if x >= 3.5 else 0)
X = chocolateratings_df.drop(['rating','company'], axis=1).copy()
X_train, X_test, y_train, y_test = train_test_split(X, y,\
                                test_size=0.33, random_state=rand)
```

```
X_train_nlp = tastes_s[X_train.index]
X_test_nlp = tastes_s[X_test.index]
```

Train C-Support Vector Classification Model

```
orig_plt_params = plt.rcParams
sns.set()
svm_mdl = svm.SVC(probability=True, gamma='auto', random_state=rand)
fitted_svm_mdl = svm_mdl.fit(X_train, y_train)
y_train_svc_pred, y_test_svc_prob, y_test_svc_pred =\
        mldatasets.evaluate_class_mdl(fitted_svm_mdl, X_train,\
                                      X_test, y_train, y_test)
```



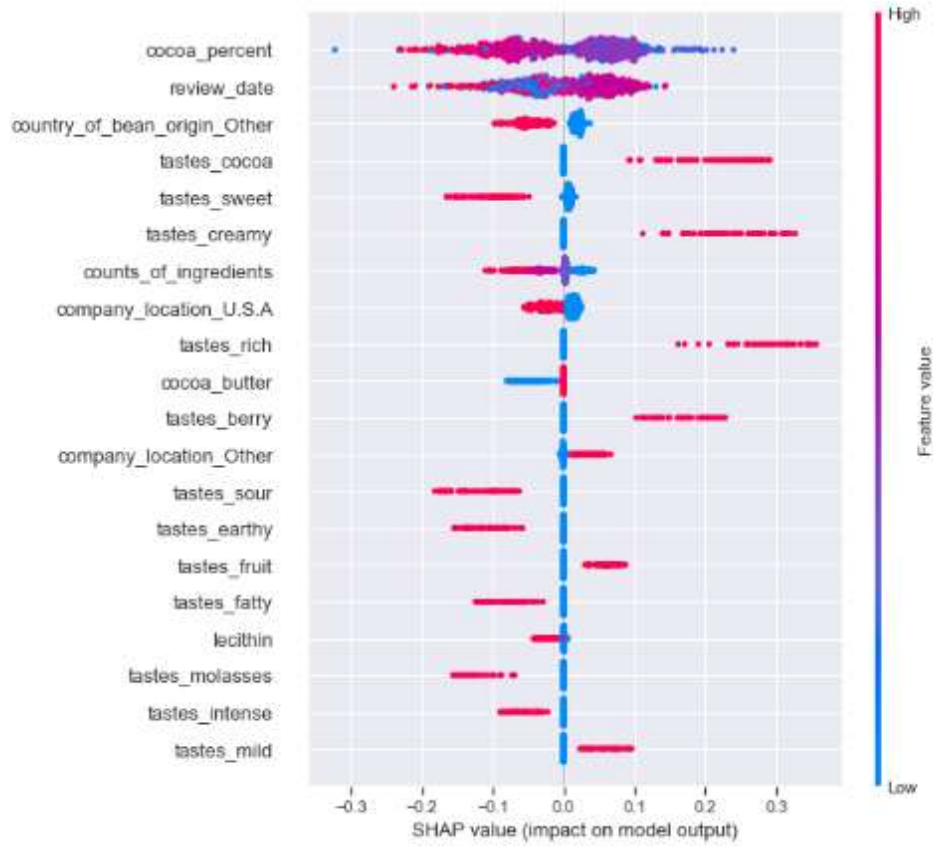

```
Accuracy_train:  0.7315        Accuracy_test:   0.6962
Precision_test:  0.6772        Recall_test:     0.4414
ROC-AUC_test:    0.7449        F1_test:         0.5344        MCC_test: 0.3399
```

```
print(svm_mdl.classes_)
```

```
array([0, 1])
```

```
np.random.seed(rand)
X_train_summary = shap.kmeans(X_train, 10)
shap_svm_explainer = shap.KernelExplainer(fitted_svm_mdl.predict_proba,\
                                          X_train_summary)
shap_svm_values_test = shap_svm_explainer.shap_values(X_test,\
                                nsamples=200, l1_reg="num_features(20)")
shap.summary_plot(shap_svm_values_test[1], X_test, plot_type="dot")
```

```
HBox(children=(FloatProgress(value=0.0, max=734.0), HTML(value='')))
```



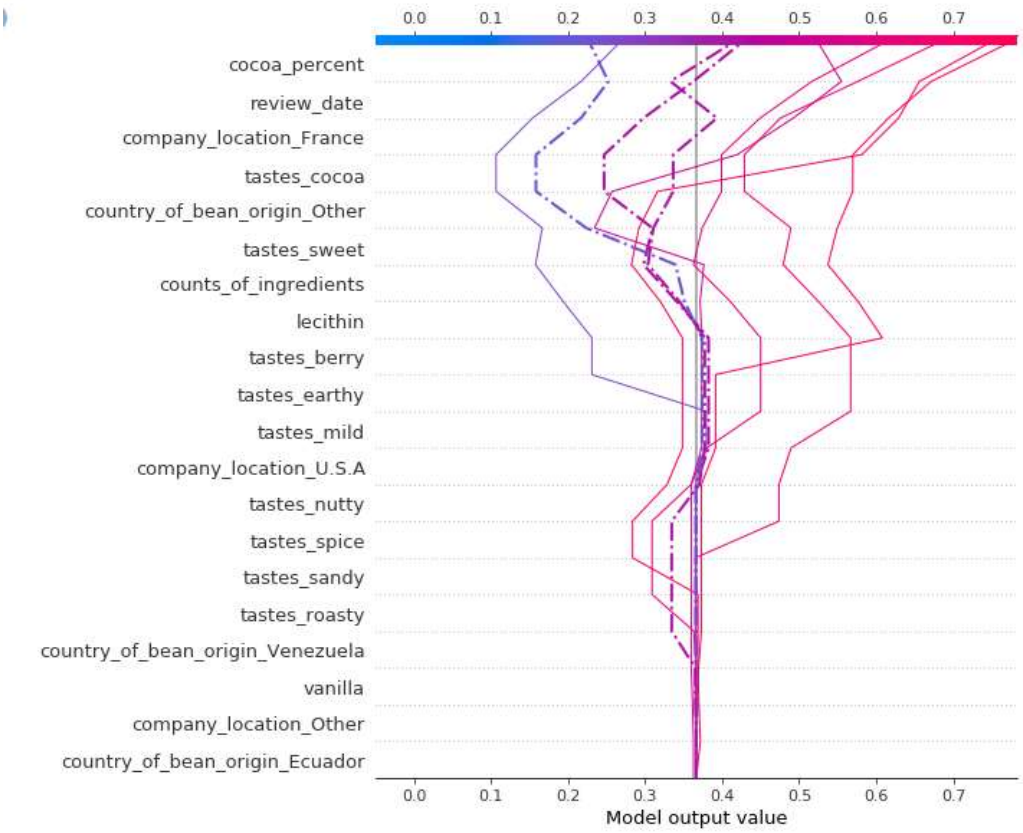

```
sample_test_idx = X_test.index.\
                    get_indexer_for([5,6,7,18,19,21,24,25,27])
```

```
expected_value = shap_svm_explainer.expected_value[1]
y_test_shap_pred = (shap_svm_values_test[1].sum(1) + expected_value) > 0.5
print(np.array_equal(y_test_shap_pred, y_test_svc_pred))
```

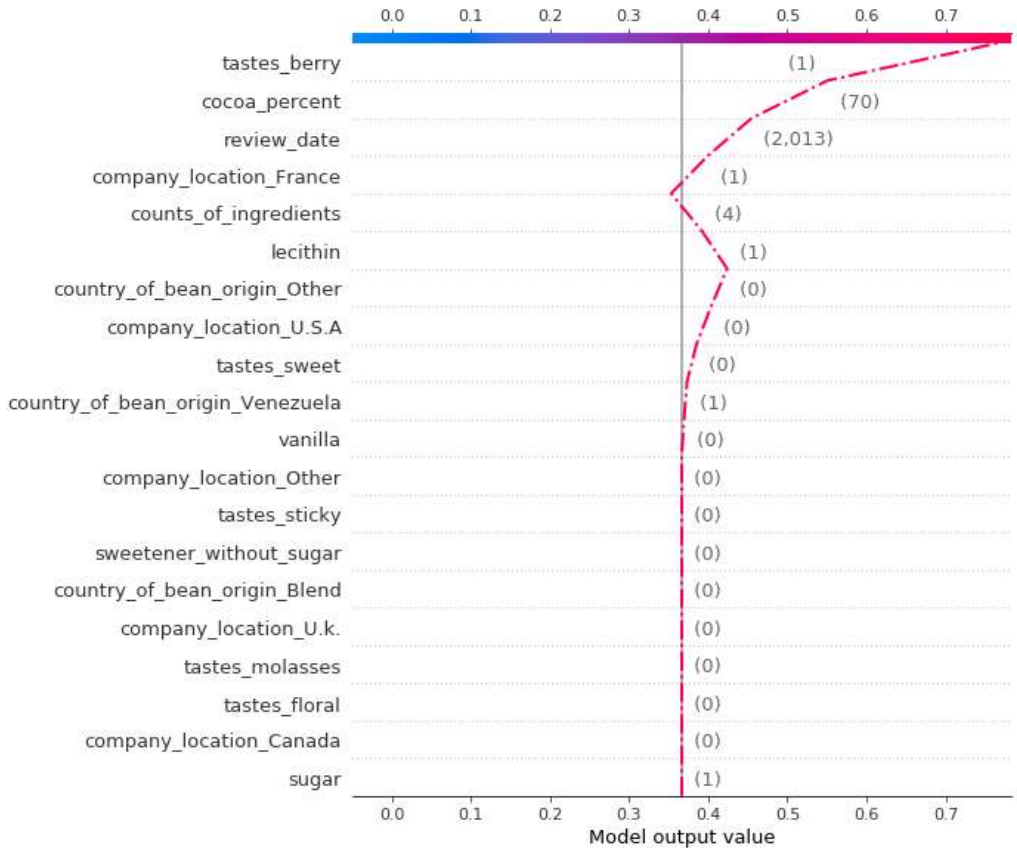
```
True
```

```
FN = (~y_test_shap_pred[sample_test_idx]) &\
     (y_test.iloc[sample_test_idx] == 1).to_numpy()
```

```
sns.reset_orig()
plt.rcParams.update(orig_plt_params)
shap.decision_plot(expected_value, shap_svm_values_test[1][sample_test_idx],\
                   X_test.iloc[sample_test_idx], highlight=FN)
```



```
shap.decision_plot(expected_value, shap_svm_values_test[1][696],\
                   X_test.iloc[696], highlight=0)
```



Local Interpretation for a single prediction at a time using Force Plot

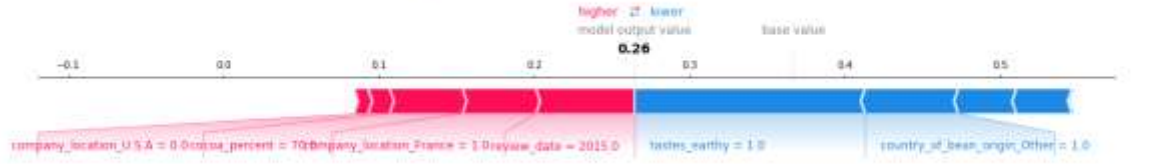
```
eval_idxs = (X_test.index==5) | (X_test.index==24)
X_test_eval = X_test[eval_idxs]
eval_compare_df = pd.concat([\
            chocolateratings_df.iloc[X_test[eval_idxs].index].rating,\
            pd.DataFrame({'y':y_test[eval_idxs]}, index=[5,24]),\
            pd.DataFrame({'y_pred':y_test_svc_pred[eval_idxs]},\
                         index=[24,5]),
            X_test_eval], axis=1).transpose()
eval_compare_df
```

	5	24
rating	4.0	2.75
y	1.0	0.00
y_pred	1.0	0.00
review_date	2013.0	2015.00
cocoa_percent	70.0	70.00
counts_of_ingredients	4.0	4.00
cocoa_butter	1.0	1.00
vanilla	0.0	0.00
lecithin	1.0	1.00
salt	0.0	0.00
sugar	1.0	1.00
sweetener_without_sugar	0.0	0.00
company_location_Canada	0.0	0.00
company_location_France	1.0	1.00
company_location_Other	0.0	0.00
company_location_U.S.A	0.0	0.00

company_location_U.k.	0.0	0.00
country_of_bean_origin_Blend	0.0	0.00
country_of_bean_origin_Dominican_republic	0.0	0.00
country_of_bean_origin_Ecuador	0.0	0.00
country_of_bean_origin_Madagascar	0.0	0.00
country_of_bean_origin_Nicaragua	0.0	0.00
country_of_bean_origin_Other	0.0	1.00
country_of_bean_origin_Peru	0.0	0.00
country_of_bean_origin_Venezuela	1.0	0.00
tastes_cocoa	0.0	0.00
tastes_rich	0.0	0.00
tastes_fatty	0.0	0.00
tastes_roasty	0.0	0.00
tastes_nutty	0.0	0.00
tastes_sweet	0.0	0.00
tastes_sandy	0.0	0.00
tastes_sour	0.0	0.00
tastes_intense	0.0	0.00
tastes_mild	0.0	0.00
tastes_fruit	0.0	0.00
tastes_sticky	0.0	0.00
tastes_earthy	0.0	1.00
tastes_spice	0.0	0.00
tastes_molasses	0.0	0.00
tastes_floral	0.0	0.00

```
shap.force_plot(expected_value, shap_svm_values_test[1][X_test.index==5],\
                X_test[X_test.index==5],  matplotlib=True)
```


```
shap.force_plot(expected_value, shap_svm_values_test[1][X_test.index==24],\
                X_test[X_test.index==24], matplotlib=True)
```



Local Interpretation for a single prediction at a time using LimeTabularExplainer

```
lime_svm_explainer = lime.lime_tabular.LimeTabularExplainer(X_test.values,\
                     feature_names=X_test.columns, categorical_features=list(range(3,44)),\
                     class_names=['Not Highly Recomm.', 'Highly Recomm.'])
```

```
lime_svm_explainer.explain_instance(X_test[X_test.index==5].values[0],\
                                    fitted_svm_mdl.predict_proba,\
                                    num_features=8).\
                                show_in_notebook(predict_proba=True)
```


```
lime_svm_explainer.explain_instance(X_test[X_test.index==24].values[0],\
                                    fitted_svm_mdl.predict_proba,\
                                    num_features=8).\
                                show_in_notebook(predict_proba=True)
```

Using LIME for Natural Language Processing

```
print(X_test_nlp)
```

```
1194                 roasty nutty rich
77       roasty oddly sweet marshmallow
121                balanced cherry choco
411                  sweet floral yogurt
1259             creamy burnt nuts woody
                         ...
327            sweet mild molasses bland
1832            intense fruity mild sour
464               roasty sour milk note
2013             nutty fruit sour floral
1190             rich roasty nutty smoke
Length: 734, dtype: object
```

```
vectorizer = TfidfVectorizer(lowercase=False)
X_train_nlp_fit = vectorizer.fit_transform(X_train_nlp)
X_test_nlp_fit = vectorizer.transform(X_test_nlp)
```

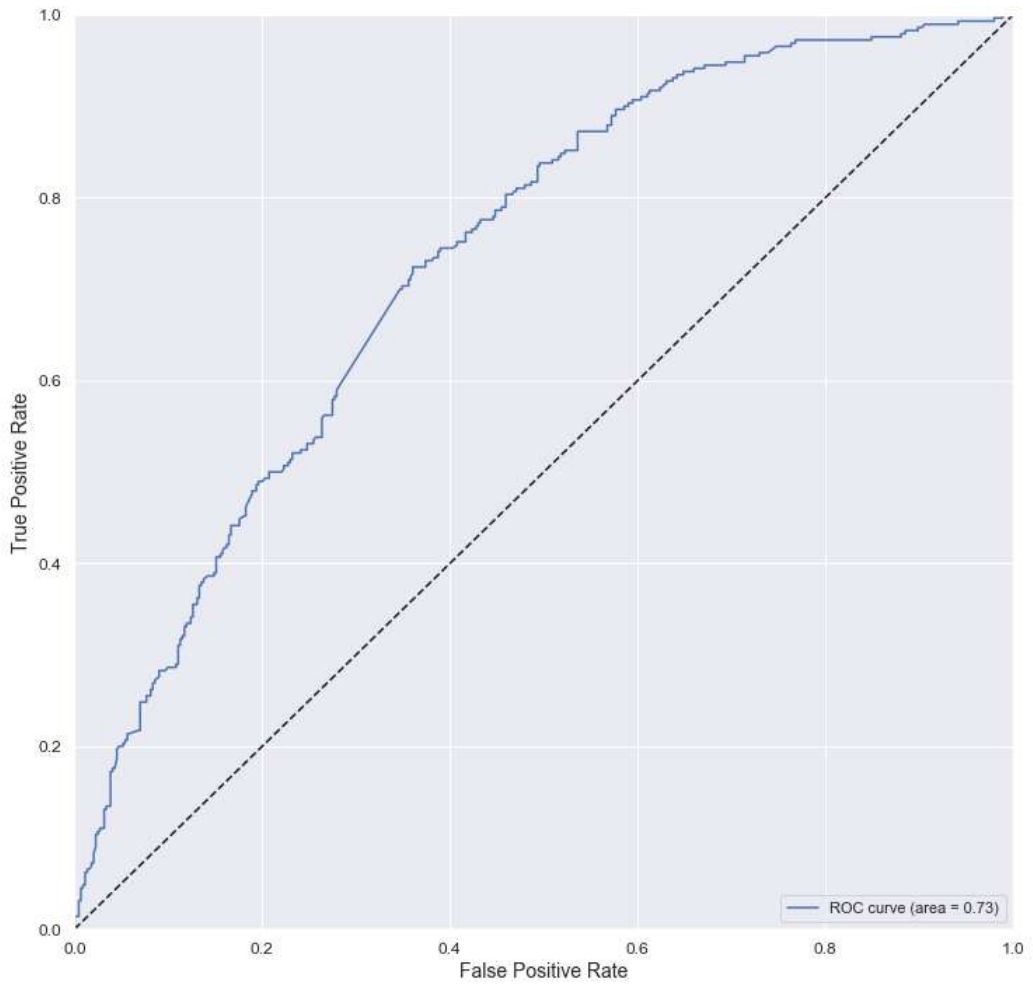
```
pd.DataFrame({'taste':vectorizer.get_feature_names(),\
              'tf-idf': np.asarray(X_test_nlp_fit[X_test_nlp.index==5].todense())[0]}).\
    sort_values(by='tf-idf', ascending=False)
```

	taste	tf-idf
305	raspberry	0.585538
259	nut	0.491542
265	oily	0.463973
64	caramel	0.447504
274	papaya	0.000000
...	...	...
136	empty	0.000000
135	edge	0.000000
134	easy	0.000000
133	easter	0.000000
415	yogurt	0.000000

416 rows × 2 columns

Train LightGBM Model

```
sns.set()
lgb_mdl = lgb.LGBMClassifier(max_depth=13, learning_rate=0.05,\
            n_estimators=100, objective='binary', random_state=rand)
fitted_lgb_mdl = lgb_mdl.fit(X_train_nlp_fit, y_train)
y_train_lgb_pred, y_test_lgb_prob, y_test_lgb_pred =\
    mldatasets.evaluate_class_mdl(fitted_lgb_mdl, X_train_nlp_fit,\
                                  X_test_nlp_fit, y_train, y_test)
```




```
Accuracy_train:  0.7953        Accuracy_test:   0.6798
Precision_test:  0.6233        Recall_test:     0.4793
ROC-AUC_test:    0.7328        F1_test:         0.5419        MCC_test: 0.3084
```

+ Code + Text

```
lgb_pipeline = make_pipeline(vectorizer, lgb_mdl)
```

```
lime_lgb_explainer = LimeTextExplainer(class_names=['Not Highly Recomm.', 'Highly Recomm.'])
```

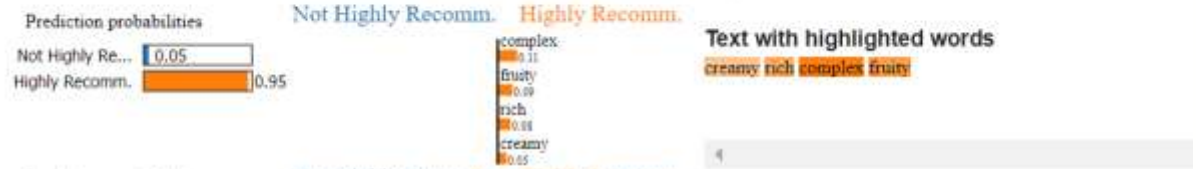
+ Code + Text

```
lime_lgb_explainer.explain_instance(X_test_nlp[X_test_nlp.index==5].values[0],\
                                    lgb_pipeline.predict_proba, num_features=4).\
                                    show_in_notebook(text=True)
```

+ Code + Text

```
lime_lgb_explainer.explain_instance(X_test_nlp[X_test_nlp.index==24].values[0],\
                                    lgb_pipeline.predict_proba, num_features=4).\
                                    show_in_notebook(text=True)
```

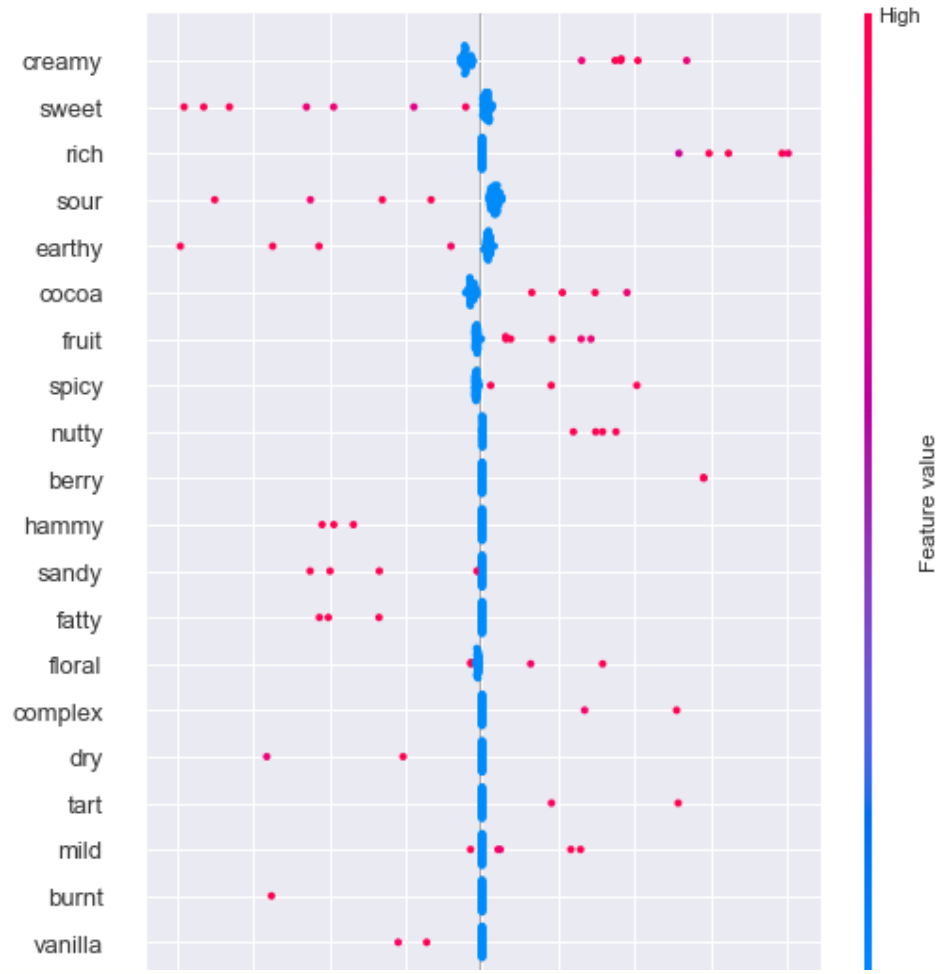

```
lime_lgb_explainer.explain_instance('creamy rich complex fruity',\
                                    lgb_pipeline.predict_proba, num_features=4).\
                                    show_in_notebook(text=True)
lime_lgb_explainer.explain_instance('sour bitter roasty molasses',\
                                    lgb_pipeline.predict_proba, num_features=4).\
                                    show_in_notebook(text=True)
lime_lgb_explainer.explain_instance('nasty disgusting gross stuff',\
                                    lgb_pipeline.predict_proba, num_features=4).\
                                    show_in_notebook(text=True)
```



```
predict_fn = lambda X: lgb_mdl.predict_proba(X)[:,1]
```

```
X_test_nlp_samp_df = pd.DataFrame(shap.sample(X_test_nlp_fit, 50).todense())
shap_lgb_explainer = shap.KernelExplainer(predict_fn,\
                          shap.kmeans(X_train_nlp_fit.todense(), 10))
shap_lgb_values_test = shap_lgb_explainer.shap_values(X_test_nlp_samp_df,\
                                              l1_reg="num_features(20)")
shap.summary_plot(shap_lgb_values_test, X_test_nlp_samp_df,\
            plot_type="dot", feature_names=vectorizer.get_feature_names())
```

```
HBox(children=(FloatProgress(value=0.0, max=50.0), HTML(value='')))
```




```
print(shap.sample(X_test_nlp, 50).to_list()[18])
```

```
woody earthy medicinal
```

```
shap.initjs()
shap.force_plot(shap_lgb_explainer.expected_value, shap_lgb_values_test[18,:],
                X_test_nlp_samp_df.iloc[18,:], feature_names=vectorizer.get_feature_names())
```

```
print(shap.sample(X_test_nlp, 50).to_list()[18])
```

```
woody earthy medicinal
```

```
shap.initjs()
shap.force_plot(shap_lgb_explainer.expected_value, shap_lgb_values_test[18,:],
                X_test_nlp_samp_df.iloc[18,:], feature_names=vectorizer.get_feature_names())
```

```
print(shap.sample(X_test_nlp, 50).to_list()[9])
```

```
intense spicy floral
```

```
shap.initjs()
shap.force_plot(shap_lgb_explainer.expected_value, shap_lgb_values_test[9,:],
                X_test_nlp_samp_df.iloc[9,:], feature_names=vectorizer.get_feature_names())
```

Chapter 7 (Ricidivism)

```
!pip install --upgrade pandas numpy scikit-learn tensorflow matplotlib seaborn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages (1.3.5)
Collecting pandas
  Downloading pandas-1.5.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.2/12.2 MB 31.5 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (1.21.6)
Collecting numpy
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 31.9 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.8/dist-packages (1.0.2)
Collecting scikit-learn
  Downloading scikit_learn-1.2.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (9.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.8/9.8 MB 19.9 MB/s eta 0:00:00
Requirement already satisfied: tensorflow in /usr/local/lib/python3.8/dist-packages (2.9.2)
Collecting tensorflow
  Downloading tensorflow-2.11.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (588.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 588.3/588.3 MB 1.2 MB/s eta 0:00:00
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (3.2.2)
Collecting matplotlib
  Downloading matplotlib-3.6.3-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (9.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.4/9.4 MB 62.2 MB/s eta 0:00:00
Requirement already satisfied: seaborn in /usr/local/lib/python3.8/dist-packages (0.11.2)
Collecting seaborn
  Downloading seaborn-0.12.2-py3-none-any.whl (293 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 293.3/293.3 KB 26.9 MB/s eta 0:00:00
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (3.1.0)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.7.3)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (4.4.0)
```

```
import math
import machine_learning_datasets as mldatasets
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import metrics
from catboost import CatBoostClassifier
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.pyplot as plt
import seaborn as sns
from alibi.utils.mapping import ohe_to_ord, ord_to_ohe
from alibi.explainers import AnchorTabular
from alibi.explainers import CEM
from alibi.explainers import CounterFactualProto
import shap
import witwidget
from witwidget.notebook.visualization import WitWidget, WitConfigBuilder
```

```
print(tf.__version__)
```

```
2.2.1
```

```
tf.compat.v1.disable_eager_execution()
print('Eager execution enabled: ', tf.executing_eagerly())
```

```
Eager execution enabled:  False
```

```
recidivism_df = mldatasets.load("recidivism-risk", prepare=True)
```

```
https://storage.googleapis.com/what-if-tool-resources/computefest2019/cox-violent-parsed_filt.csv downloaded to
1 dataset files found in /Users/smasis/Documents/OTHER/InterpretableMLBook/programming/Chapter7/data folder
parsing /Users/smasis/Documents/OTHER/InterpretableMLBook/programming/Chapter7/data/cox-violent-parsed_filt.csv
```

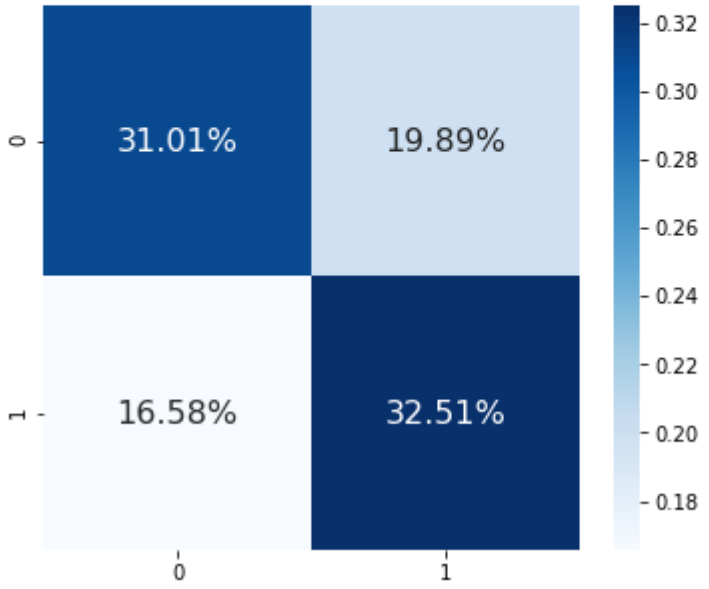
re should be almost 15,000 records and 23 columns. We can verify this was the case with `info()`:

```
recidivism_df.info()
```

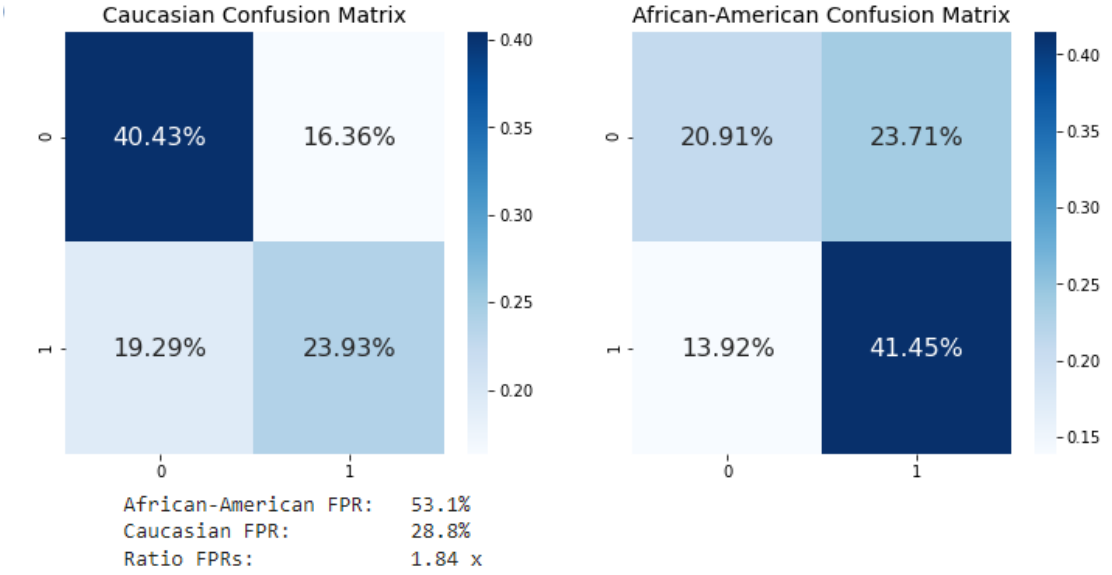
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14788 entries, 0 to 18315
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   age                    14788 non-null  int8
 1   juv_fel_count          14788 non-null  int8
 2   juv_misd_count         14788 non-null  int8
 3   juv_other_count        14788 non-null  int64
 4   priors_count           14788 non-null  int8
 5   is_recid               14788 non-null  int8
 6   sex_Female             14788 non-null  uint8
 7   sex_Male               14788 non-null  uint8
 8   race_African-American  14788 non-null  uint8
 9   race_Asian             14788 non-null  uint8
 10  race_Caucasian         14788 non-null  uint8
 11  race_Hispanic          14788 non-null  uint8
 12  race_Native American   14788 non-null  uint8
 13  race_Other             14788 non-null  uint8
 14  c_charge_degree_(F1)   14788 non-null  uint8
 15  c_charge_degree_(F2)   14788 non-null  uint8
 16  c_charge_degree_(F3)   14788 non-null  uint8
 17  c_charge_degree_(F7)   14788 non-null  uint8
 18  c_charge_degree_(M1)   14788 non-null  uint8
 19  c_charge_degree_(M2)   14788 non-null  uint8
 20  c_charge_degree_(MO3)  14788 non-null  uint8
 21  c_charge_degree_Other  14788 non-null  uint8
 22  compas_score           14788 non-null  int64
dtypes: int64(2), int8(5), uint8(16)
memory usage: 649.9 KB
```

```
cf_matrix = metrics.confusion_matrix(recidivism_df.is_recid,\
                                     recidivism_df.compas_score)
plt.figure(figsize=(6, 5))
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True,\
            fmt='.2%', cmap='Blues', annot_kws={'size':16})
```

```
<AxesSubplot:>
```



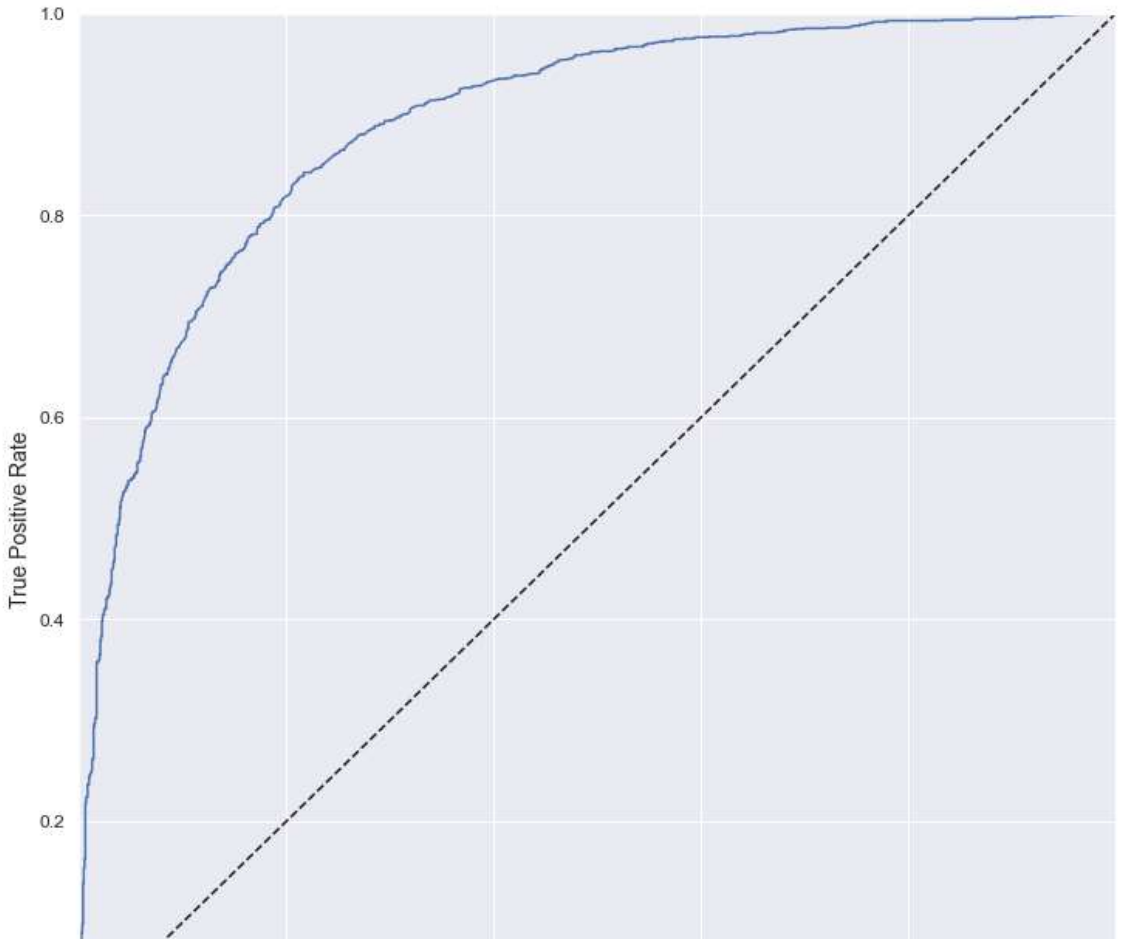
```
recidivism_c_df = recidivism_df[recidivism_df['race_Caucasian'] == 1]
recidivism_aa_df = recidivism_df[recidivism_df['race_African-American'] == 1]

_ = mldatasets.\
      compare_confusion_matrices(recidivism_c_df.is_recid,
                                 recidivism_c_df.compas_score,\
                                 recidivism_aa_df.is_recid,\
                                 recidivism_aa_df.compas_score,\
                                 'Caucasian', 'African-American', compare_fpr=True)
```

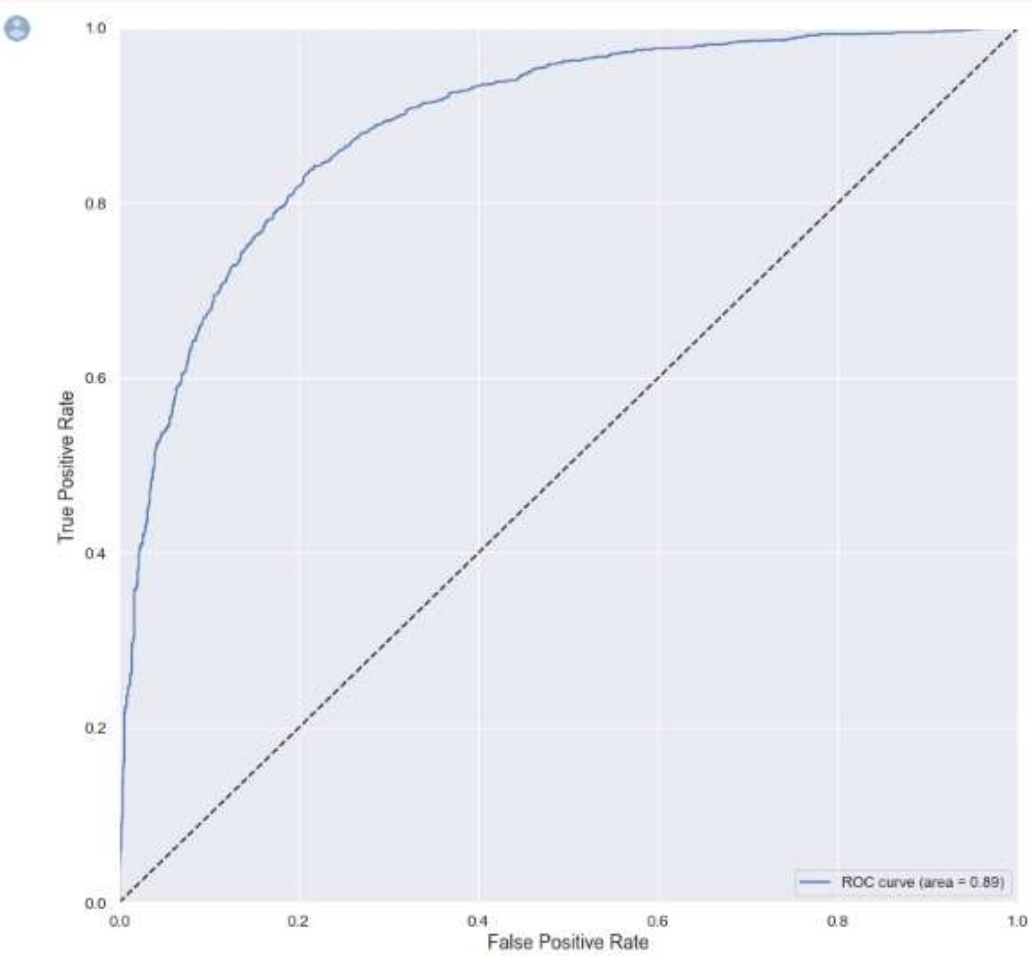



```
African-American FPR:    53.1%
Caucasian FPR:           28.8%
Ratio FPRs:              1.84 x
```

```
rand = 9
np.random.seed(rand)
tf.random.set_seed(rand)

y = recidivism_df['compas_score']
X = recidivism_df.drop(['compas_score', 'is_recid'], axis=1).copy()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=rand)
```

```
orig_plt_params = plt.rcParams
sns.set()
cb_mdl = CatBoostClassifier(iterations=500, learning_rate=0.5, depth=8)
fitted_cb_mdl = cb_mdl.fit(X_train, y_train, verbose=False)
y_train_cb_pred, y_test_cb_prob, y_test_cb_pred =\
        mldatasets.evaluate_class_mdl(fitted_cb_mdl, X_train,\
                                      X_test, y_train, y_test)
```





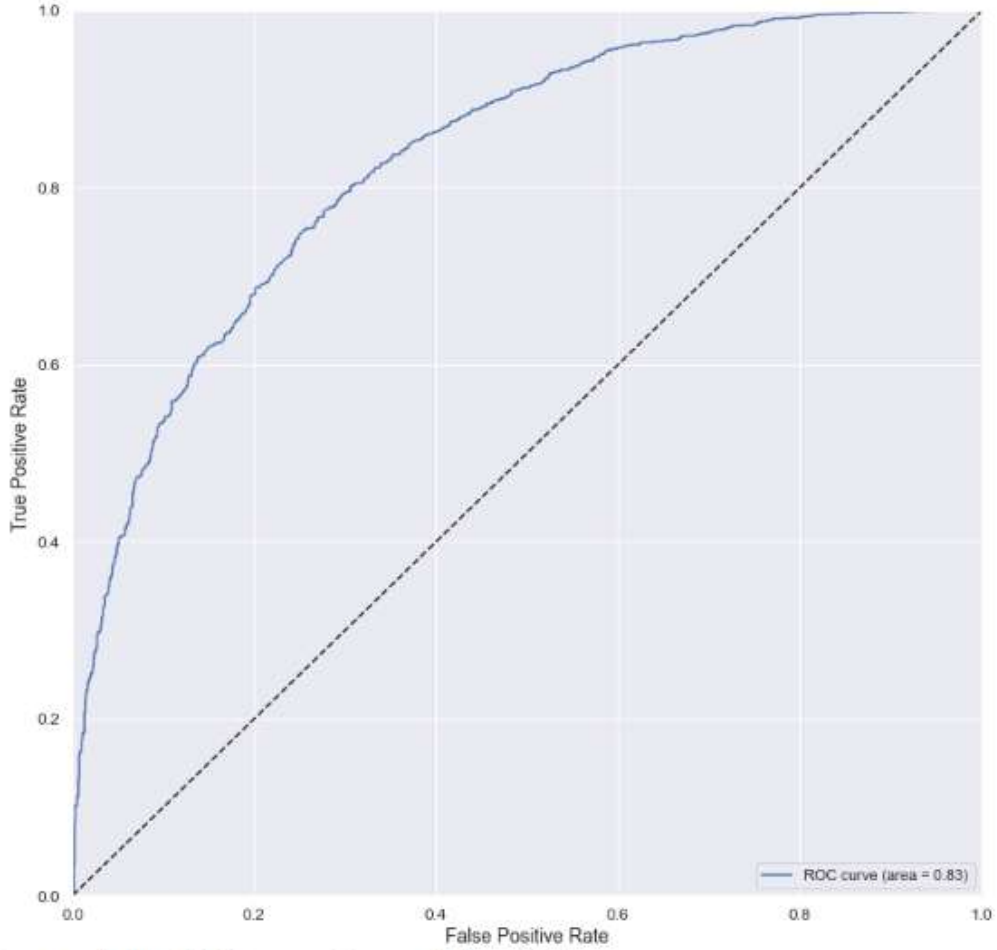

```
Accuracy_train:  0.8793        Accuracy_test:   0.8087
Precision_test:  0.8294        Recall_test:     0.8085
ROC-AUC_test:    0.8904        F1_test:         0.8188        MCC_test: 0.6164
```

Neural Network

```
fitted_nn_mdl = keras.Sequential([
    tf.keras.Input(shape=[len(X_train.keys())]),
    layers.Dense(7, activation='relu'),
    layers.Dense(1, activation='sigmoid')
  ])
fitted_nn_mdl.compile(loss='mean_squared_error', optimizer='adam')
nn_history = fitted_nn_mdl.fit(X_train.values, y_train.values, epochs=12,\
                               batch_size=32, validation_split=0.2, verbose=0)
y_train_nn_pred, y_test_nn_prob, y_test_nn_pred =\
        mldatasets.evaluate_class_mdl(fitted_nn_mdl, X_train,\
                                      X_test, y_train, y_test)
```

```
WARNING:tensorflow:From /opt/anaconda3/lib/python3.8/site-packages/tensorflow/python/ops/resou
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
```

```
WARNING:tensorflow:From /opt/anaconda3/lib/python3.8/site-packages/tensorflow/python/ops/resource_variable_c
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
```




```
Accuracy_train:  0.7527        Accuracy_test:   0.7471
Precision_test:  0.7653        Recall_test:     0.7604
ROC-AUC_test:    0.8320        F1_test:         0.7628        MCC_test: 0.4920
```

```
idx1 = 5231
idx2 = 2726
idx3 = 10127
eval_idxs = X_test.index.isin([idx1, idx2, idx3])
X_test_evals = X_test[eval_idxs]
eval_compare_df = pd.concat([\
            pd.DataFrame({'y':y_test[eval_idxs]},\
                         index=[idx3, idx2, idx1]),\
            pd.DataFrame({'y_pred':y_test_cb_pred[eval_idxs]},\
                         index=[idx3, idx2, idx1]),\
            X_test_evals], axis=1).transpose()
eval_compare_df
```

	10127	2726	5231
y	0	0	1
y_pred	0	0	1
age	24	23	23
juv_fel_count	0	0	0
juv_misd_count	0	0	0
juv_other_count	0	0	0
priors_count	2	2	2
sex_Female	0	0	0
sex_Male	1	1	1
race_African-American	0	0	1
race_Asian	0	0	0
race_Caucasian	1	0	0
race_Hispanic	0	1	0
race_Native American	0	0	0
race_Other	0	0	0

race_Native American	0	0	0
race_Other	0	0	0
c_charge_degree_(F1)	0	0	0
c_charge_degree_(F2)	0	0	0
c_charge_degree_(F3)	0	1	0
c_charge_degree_(F7)	0	0	1
c_charge_degree_(M1)	1	0	0
c_charge_degree_(M2)	0	0	0
c_charge_degree_(MO3)	0	0	0
c_charge_degree_Other	0	0	0

```
class_names = ['Low Risk', 'Medium/High Risk']
```

```
X_test_eval = np.expand_dims(X_test.values[X_test.\
                                           index.get_loc(idx1)], axis=0)
print(X_test_eval)
```

```
[[23  0  0  0  2  0  1  1  0  0  0  0  0  0  0  0  1  0  0  0  0]]
```

```
cat_vars_ohe = {5: 2, 7: 6, 13: 8}
print(ohe_to_ord(X_test_eval, cat_vars_ohe)[0])
```

```
[[23  0  0  0  2  1  0  3]]
```

```
category_map = {
    5: ['Female', 'Male'],
    6: ['African-American', 'Asian', 'Caucasian',\
        'Hispanic', 'Native American', 'Other'],
    7: ['Felony 1st Degree', 'Felony 2nd Degree',\
        'Felony 3rd Degree', 'Felony 7th Degree',\
        'Misdemeanor 1st Degree', 'Misdemeanor 2nd Degree',\
        'Misdemeanor 3rd Degree', 'Other Charge Degree']
}
feature_names = ['age', 'juv_fel_count', 'juv_misd_count',\
                 'juv_other_count', 'priors_count',
                 'sex', 'race', 'c_charge_degree']
```

```
 category_map_ohe = {5: ['Not Female', 'Female'], 6: ['Not Male', 'Male'],\
         7:['Not African American', 'African American'],\
         8:['Not Asian', 'Asian'], 9:['Not Caucasian', 'Caucasian'],\
         10:['Not Hispanic', 'Hispanic'],\
         11:['Not Native American', 'Native American'],\
         12:['Not Other Race', 'Other Race'],\
         13:['Not Felony 1st Level', 'Felony 1st Level'],\
         14:['Not Felony 2nd Level', 'Felony 2nd Level'],\
         15:['Not Felony 3rd Level', 'Felony 3rd Level'],\
         16:['Not Felony 7th Level', 'Felony 7th Level'],\
         17:['Not Misdemeanor 1st Deg', 'Misdemeanor 1st Deg'],\
         18:['Not Misdemeanor 2nd Deg', 'Misdemeanor 2nd Deg'],\
         19:['Not Misdemeanor 3rd Deg', 'Misdemeanor 3rd Deg'],\
         20:['Not Other Charge Degree', 'Other Charge Degree']}
```

```
predict_cb_fn = lambda x: fitted_cb_mdl.predict_proba(x)
anchor_cb_explainer = AnchorTabular(predict_cb_fn, X_train.columns,\
                                    categorical_names=category_map_ohe)
anchor_cb_explainer.fit(X_train.values)
```

```
AnchorTabular(meta={
  'name': 'AnchorTabular',
  'type': ['blackbox'],
  'explanations': ['local'],
  'params': {'seed': None, 'disc_perc': (25, 50, 75)}}
)
```

```
print('Prediction: %s' %\
      class_names[anchor_cb_explainer.predictor(X_test.loc[idx1].values)[0]])
```

```
Prediction: Medium/High Risk
```

```
anchor_cb_explanation =\
    anchor_cb_explainer.explain(X_test.loc[idx1].values, threshold=0.85,\
                                seed=rand)
print('Anchor: %s' % (' AND\r\n\t'.join(anchor_cb_explanation.anchor)))
print('Precision: %.3f' % anchor_cb_explanation.precision)
print('Coverage: %.3f' % anchor_cb_explanation.coverage)
```

```
Anchor: age <= 25.00 AND
        priors_count > 0.00 AND
        race_African-American = African American
Precision: 0.891
Coverage: 0.290
```

```
anchor_cb_explanation =\
    anchor_cb_explainer.explain(X_test.loc[idx1].values, threshold=0.9,\
                                seed=rand)
print('Anchor: %s' % (' AND\r\n\t'.join(anchor_cb_explanation.anchor)))
print('Precision: %.3f' % anchor_cb_explanation.precision)
print('Coverage: %.3f' % anchor_cb_explanation.coverage)
```

```
Anchor: age <= 25.00 AND
        priors_count > 0.00 AND
        race_African-American = African American AND
        race_Other = Not Other Race AND
        sex_Female = Not Female AND
        c_charge_degree_(M2) = Not Misdemeanor 2nd Deg AND
        race_Hispanic = Not Hispanic AND
        c_charge_degree_(M1) = Not Misdemeanor 1st Deg
Precision: 0.900
Coverage: 0.290
```

```
predict_nn_fn = lambda x: np.concatenate((1 - fitted_nn_mdl.predict(x),\
                                          fitted_nn_mdl.predict(x)), axis=1)
anchor_nn_explainer = AnchorTabular(predict_nn_fn, X_train.columns,\
                                    categorical_names=category_map_ohe)
anchor_nn_explainer.fit(X_train.values)
```

```
AnchorTabular(meta={
  'name': 'AnchorTabular',
  'type': ['blackbox'],
  'explanations': ['local'],
  'params': {'seed': None, 'disc_perc': (25, 50, 75)}}
)
```

```
anchor_nn_explanation =\
    anchor_nn_explainer.explain(X_test.loc[idx3].values, threshold=0.85,\
                                seed=rand)
print('Anchor: %s' % (' AND\r\n\t'.join(anchor_nn_explanation.anchor)))
print('Precision: %.3f' % anchor_nn_explanation.precision)
print('Coverage: %.3f' % anchor_nn_explanation.coverage)
```

```
Anchor: priors_count <= 2.00 AND
        c_charge_degree_(M1) = Misdemeanor 1st Deg
Precision: 0.906
Coverage: 0.578
```

```
anchor_nn_explanation =\
    anchor_nn_explainer.explain(X_test.loc[idx2].values, threshold=0.85,\
                                seed=rand)
print('Anchor: %s' % (' AND\r\n\t'.join(anchor_nn_explanation.anchor)))
print('Precision: %.3f' % anchor_nn_explanation.precision)
print('Coverage: %.3f' % anchor_nn_explanation.coverage)
```

```
Anchor: priors_count <= 2.00 AND
        race_African-American = Not African American AND
        race_Caucasian = Not Caucasian
Precision: 0.896
Coverage: 0.578
```

COUNTERFACTUAL Explanations

```
feature_range = (X_train.values.min(axis=0).reshape(1,21).astype(np.float32),\
                 X_train.values.max(axis=0).reshape(1,21).astype(np.float32))
print(feature_range)
```

```
(array([[18.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.]], dtype=float32), array([[96., 20., 13., 11., 38.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]], dtype=float32))
```

+ Code + Text

```
cf_nn_explainer = CounterFactualProto(predict_nn_fn,
                                      X_test_eval.shape,
                                      max_iterations=100,
                                      feature_range=feature_range,
                                      beta=.1, theta=5,
                                      use_kdtree=True
                                     )
cf_nn_explainer.fit(X_test.values, d_type='abdm-mvdm')
```

```
No encoder specified. Using k-d trees to represent class prototypes.
CounterFactualProto(meta={
  'name': 'CounterFactualProto',
  'type': ['blackbox', 'tensorflow', 'keras'],
  'explanations': ['local'],
  'params': {
            'kappa': 0.0,
            'beta': 0.1,
            'feature_range': (array([[18.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.]], dtype=float32), array([[96., 20., 13., 11., 38.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]], dtype=float32)),
            'gamma': 0.0,
            'theta': 5,
            'cat_vars': None,
            'ohe': False,
            'use_kdtree': True,
            'learning_rate_init': 0.01,
            'max_iterations': 100,
            'c_init': 10.0,
            'c_steps': 10,
            'eps': (0.001, 0.001),
            'clip': (-1000.0, 1000.0),
            'update_num_grad': 1,
            'write_dir': None,
            'shape': (1, 21),
            'is_model': False,
            'is_model_keras': False,
            'is_ae': False,
            'is_ae_keras': False,
            'is_enc': False,
            'is_enc_keras': False,
            'enc_or_kdtree': True,
            'is_cat': False,
            'trustscore_kwargs': None,
            'd_type': 'abdm-mvdm',
            'w': None,
            'disc_perc': (25, 50, 75),
            'standardize_cat_vars': False,
            'smooth': 1.0,
            'center': True,
            'update_feature_range': True}
        }
```

```
cf_nn_explanation = cf_nn_explainer.explain(X_test_eval)
mldatasets.describe_cf_instance(X_test_eval, cf_nn_explanation, class_names,\
                  cat_vars_ohe, category_map, feature_names)
```

```
Instance Outcomes and Probabilities
-----------------------------------------------
       original:  Medium/High Risk
                  [0.46732193 0.53267807]
 counterfactual:  Low Risk
                  [0.50025815 0.49974185]

Categorical Feature Counterfactual Perturbations
-----------------------------------------------
                    sex:  Male  -->  Female
                   race:  African-American  -->  Asian
         c_charge_degree:  Felony 7th Degree  -->  Felony 1st Degree

Numerical Feature Counterfactual Perturbations
-----------------------------------------------
            priors_count:  2.00  -->  1.90
```

Configuring WIT Tool

```
shap_cb_explainer = shap.TreeExplainer(fitted_cb_mdl)
```

```
test_df = recidivism_df.loc[y_test.index]
test_np = test_df.values
cols_l = test_df.columns
delcol_idx = [cols_l.get_loc("is_recid"),\
              cols_l.get_loc("compas_score")]
```

```
def custom_predict_with_shap(examples_np):
    #For shap values we only need same features
    #that were used for training
    inputs_np = np.delete(np.array(examples_np),\
                          delcol_idx, axis=1)

    #Get the model's class predictions
    preds = predict_cb_fn(inputs_np)

    #With test data generate SHAP values which converted
    #to a list of dictionaries format
    keepcols_l = [c for i, c in enumerate(cols_l)\
                  if i not in delcol_idx]
    shap_output = shap_cb_explainer.shap_values(inputs_np)
    attributions = []
    for shap in shap_output:
        attrs = {}
        for i, col in enumerate(keepcols_l):
            attrs[col] = shap[i]
        attributions.append(attrs)
```

```
        #Get the model's class predictions
        preds = predict_cb_fn(inputs_np)

        #With test data generate SHAP values which converted
        #to a list of dictionaries format
        keepcols_l = [c for i, c in enumerate(cols_l)\
                      if i not in delcol_idx]
        shap_output = shap_cb_explainer.shap_values(inputs_np)
        attributions = []
        for shap in shap_output:
            attrs = {}
            for i, col in enumerate(keepcols_l):
                attrs[col] = shap[i]
            attributions.append(attrs)

        #Prediction function must output
        #predictions/attributions in dictionary
        output = {'predictions': preds,\
                  'attributions': attributions}
        return output
```

```
print(y_test.index.get_loc(5231))
```

```
2910
```

```
wit_config_builder = WitConfigBuilder(\
                    test_np.tolist(), feature_names=cols_l.tolist()
                ).set_custom_predict_fn(custom_predict_with_shap).\
                set_target_feature("is_recid").set_label_vocab(class_names)
WitWidget(wit_config_builder, height=800)
```

```
WitWidget(config={'model_type': 'classification', 'label_vocab': ['Low Risk', 'Medium/High Risk'], 'feature_na…
```

Autoencoder

```
input_layer = tf.keras.Input(shape=(21))
encoder = tf.keras.layers.Dense(10, activation='relu')(input_layer)
bottleneck = tf.keras.layers.Dense(3, activation='relu')(encoder)
decoder = tf.keras.layers.Dense(10, activation='relu')(bottleneck)
output_layer = tf.keras.layers.Dense(21, activation='linear')(decoder)
autoencoder_mdl = tf.keras.Model(input_layer, output_layer)
autoencoder_mdl.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 21)]              0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                220
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 33
_________________________________________________________________
dense_4 (Dense)              (None, 10)                40
_________________________________________________________________
dense_5 (Dense)              (None, 21)                231
=================================================================
Total params: 524
Trainable params: 524
Non-trainable params: 0
_________________________________________________________________
```

```
autoencoder_mdl.compile(loss='mean_squared_error', optimizer='adam')
autoencoder_history = autoencoder_mdl.fit(X_train.values, X_train.values, epochs=16,\
                              batch_size=32, validation_split=0.2, verbose=0)
```

FIT CEM Explainers

```
cem_nn_explainer_pn = CEM(predict_nn_fn, 'PN',\
                          X_test_eval.shape,\
                          feature_range=feature_range,\
                          max_iterations=100, gamma=100,\
                          ae_model=autoencoder_mdl)
cem_nn_explainer_pn.fit(X_train.values, no_info_type='median')
cem_nn_explanation_pn = cem_nn_explainer_pn.explain(X_test_eval,\
                                                    verbose=False)
print("%s -> %s" % (class_names[cem_nn_explanation_pn.X_pred],\
                    class_names[cem_nn_explanation_pn.PN_pred]))
print("Probabilities: %s" % predict_nn_fn(cem_nn_explanation_pn.PN)[0])
print("Values: %s" % cem_nn_explanation_pn.PN[0])
```

```
Medium/High Risk -> Low Risk
Probabilities: [0.5021912 0.4978088]
Values: [2.3000000e+01 0.0000000e+00 0.0000000e+00 0.0000000e+00 2.0000000e+00
 0.0000000e+00 9.9445111e-01 7.6429874e-01 0.0000000e+00 1.2006172e-01
 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
 0.0000000e+00 1.0000000e+00 2.1608792e-02 0.0000000e+00 0.0000000e+00
 0.0000000e+00]
```

```
cem_nn_explainer_pp = CEM(predict_nn_fn, 'PP',\
                          X_test_eval.shape,\
                          feature_range=feature_range,\
                          max_iterations=100, gamma=100,\
                          ae_model=autoencoder_mdl)
cem_nn_explainer_pp.fit(X_train.values, no_info_type='median')
cem_nn_explanation_pp = cem_nn_explainer_pp.explain(X_test_eval,\
                                                    verbose=False)
print("%s -> %s" % (class_names[cem_nn_explanation_pp.X_pred],\
                    class_names[cem_nn_explanation_pp.PP_pred]))
print("Probabilities: %s" % predict_nn_fn(cem_nn_explanation_pp.PP)[0])
print("Values: %s" % cem_nn_explanation_pp.PP[0])
```

```
Medium/High Risk -> Medium/High Risk
Probabilities: [0.29961264 0.70038736]
Values: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

CEM Explanations

```
salients_df = pd.DataFrame({'Feature': X_test.columns,\
                            'x': cem_nn_explanation_pn.X[0],\
                            'PN': cem_nn_explanation_pn.PN[0],\
                            'PN-x': cem_nn_explanation_pn.PN[0]-\
                                    cem_nn_explanation_pn.X[0],\
                            'PP': cem_nn_explanation_pp.PP[0]})
salients_df = salients_df[(salients_df.PP != 0) |\
                          (salients_df.PN != 0)]
salients_df
```

	Feature	x	PN	PN-x	PP
0	age	23	23.000000	0.000000	0.0
4	priors_count	2	2.000000	0.000000	0.0
6	sex_Male	1	0.994451	-0.005549	0.0
7	race_African-American	1	0.764299	-0.235701	0.0
9	race_Caucasian	0	0.120062	0.120062	0.0
16	c_charge_degree_(F7)	1	1.000000	0.000000	0.0
17	c_charge_degree_(M1)	0	0.021609	0.021609	0.0

Chapter 8 (Fruit Classifier):

Part 1:

Installing the Libraries

These are all already installed on Google Colab by default so install only if running elsewhere (and ***not already installed***).

```
[ ] !pip install --upgrade numpy pandas scikit-learn tensorflow Keras matplotlib seaborn opencv-python
```

Install these if running on Google Colab or ***not already installed***

```
[ ] !pip install --upgrade machine-learning-datasets
    !pip install --upgrade tf-explain tf-keras-vis
```

Loading the Libraries

```
[ ] import math
    import os
    import machine_learning_datasets as mldatasets
    import numpy as np
    import pandas as pd
    from sklearn import preprocessing, metrics
    import tensorflow as tf
    from tensorflow import keras
    import matplotlib.pyplot as plt
    from matplotlib import cm
    import seaborn as sns
    import cv2
    from keras.utils.data_utils import get_file
    %%capture
    from tf_explain.core.activations import ExtractActivations
    from tf_keras_vis.activation_maximization import ActivationMaximization
    from tf_keras_vis.saliency import Saliency
    from tf_keras_vis.utils import normalize
    from tf_keras_vis.gradcam import GradcamPlusPlus
    from tf_explain.core.integrated_gradients import IntegratedGradients
```

Understanding and Preparing the Data

```
[ ] X_train, X_test, X_val, X_val_orig, y_train, y_test, y_val, y_val_orig =\
                                mldatasets.load("fruits-360", prepare=True)
```

```
https://github.com/fenilgmehta/interpretable-machine-learning-with-Python/archive/master.zip downloaded to /home/sarg/EXTERNAL/PROJECTS/InterpretableMLBook/programming/Chapter6/data/fruits-360_abbrev.zip
/home/sarg/EXTERNAL/PROJECTS/InterpretableMLBook/programming/Chapter6/data/fruits-360_abbrev.zip unzipped to /home/sarg/EXTERNAL/PROJECTS/InterpretableMLBook/programming/Chapter6/data/fruits-360_abbrev
10653 dataset files found in /home/sarg/EXTERNAL/PROJECTS/InterpretableMLBook/programming/Chapter6/data/fruits-360_abbrev folder
```

```
[ ] print('X_train:\t\t%s' % (X_train.shape,))
    print('X_test:\t\t%s' % (X_test.shape,))
    print('X_val:\t\t%s' % (X_val.shape,))
    print('X_val_orig:\t%s' % (X_val_orig.shape,))
    print('y_train:\t%s' % (y_train.shape,))
    print('y_test:\t\t%s' % (y_test.shape,))
    print('y_val:\t\t%s' % (y_val.shape,))
    print('y_val_orig:\t%s' % (y_val_orig.shape,))
```

```
X_train:        (7872, 100, 100, 3)
X_test:         (2613, 100, 100, 3)
X_val:          (44, 100, 100, 3)
X_val_orig:     (44, 400, 400, 3)
y_train:        (7872, 1)
y_test:         (2613, 1)
y_val:          (44, 1)
y_val_orig:     (44, 1)
```

Data Preparation

```
[ ] X_train = X_train.astype('float32')/255
    X_test = X_test.astype('float32')/255
    X_val = X_val.astype('float32')/255
```

```
[ ] ohe = preprocessing.OneHotEncoder(sparse=False)
    ohe.fit(y_train)
    fruits_l = ohe.categories_[0].tolist()
    print(fruits_l)
```

Data Preparation

```
[ ] X_train = X_train.astype('float32')/255
    X_test = X_test.astype('float32')/255
    X_val = X_val.astype('float32')/255
```

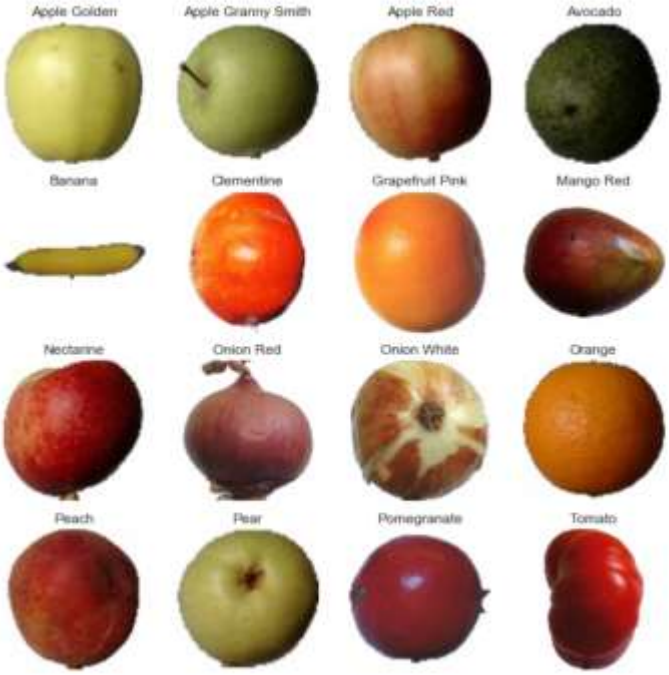
```
[ ] ohe = preprocessing.OneHotEncoder(sparse=False)
    ohe.fit(y_train)
    fruits_l = ohe.categories_[0].tolist()
    print(fruits_l)
```

```
['Apple Golden', 'Apple Granny Smith', 'Apple Red', 'Avocado', 'Banana', 'Clementine', 'Grapefruit Pink', 'Mango Red', 'Nectarine', 'Onion Red', 'Onion White', 'Orange', 'Peach', 'Pear', 'Pomegranate', 'Tomato']
```

```
[ ] rand = 5
    os.environ['PYTHONHASHSEED']=str(rand)
    np.random.seed(rand)
    tf.random.set_seed(rand)
```

Inspect Data

```
[ ] plt.subplots(figsize=(10,10))
    for f, fruit in zip(["Apple Golden","Apple Granny Smith","Apple Red","Avocado"], fruits_l):
        plt.subplot(4, 4, f+1)
        plt.title(fruits_l[f], fontsize=12)
        idx = np.random.choice(np.where(y_test[:,f] == fruit)[0], 1)[0]
        plt.imshow(X_test[idx], interpolation='spline16')
        plt.axis('off')
    plt.show()
```




```
[ ]  plt.subplots(figsize=(10,10))
     for f, fruit in zip([*range(len(fruits_l))], fruits_l):
         plt.subplot(4, 4, f+1)
         plt.title(fruits_l[f], fontsize=12)
         idx = np.random.choice(np.where(y_val[:,0] == fruit)[0], 1)[0]
         plt.imshow(X_val[idx], interpolation='spline16')
         plt.axis("off")
     plt.show()
```

Loading the CNN Model

```
model_path = get_file('CNN_fruits_final.h5',\
                      'https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python/blob/master/models/CNN_fruits_final.h5?raw=true')
cnn_fruits_mdl = keras.models.load_model(model_path)
cnn_fruits_mdl.summary()
```

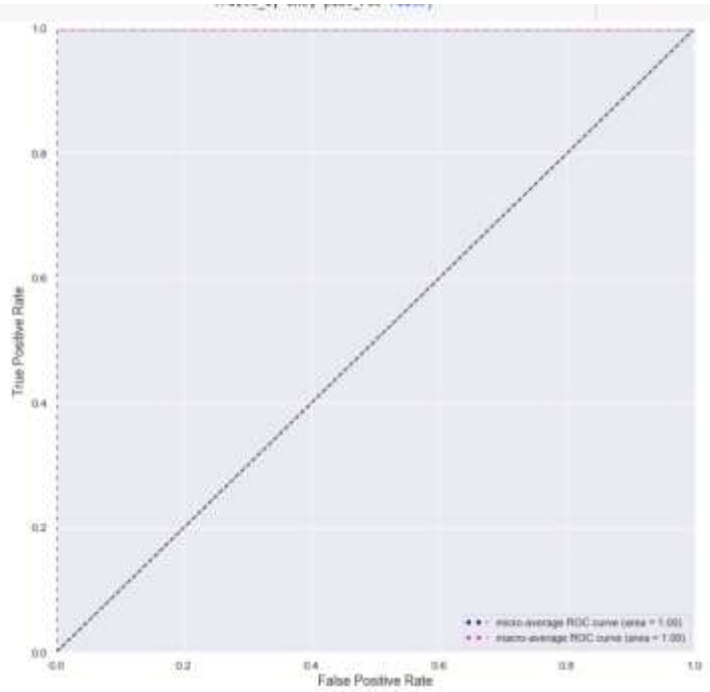
```
Downloading data from https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python/blob/master/models/CNN_fruits_final.h5?raw=true
4268032/4267608 [==============================] - 1s 1us/step
Model: "CNN_fruits"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 99, 99, 16)        208
_________________________________________________________________
maxpool2d_1 (MaxPooling2D)   (None, 49, 49, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 48, 48, 32)        8000
_________________________________________________________________
maxpool2d_2 (MaxPooling2D)   (None, 24, 24, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 23, 23, 64)        8256
_________________________________________________________________
maxpool2d_3 (MaxPooling2D)   (None, 11, 11, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 10, 10, 128)       32896
_________________________________________________________________
maxpool2d_4 (MaxPooling2D)   (None, 5, 5, 128)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 5, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 3200)              0
_________________________________________________________________
dense_1 (Dense)              (None, 150)               480150
_________________________________________________________________
dropout_2 (Dropout)          (None, 150)               0
_________________________________________________________________
dense_2 (Dense)              (None, 16)                2416
=================================================================
Total params: 531,006
Trainable params: 531,006
Non-trainable params: 0
```

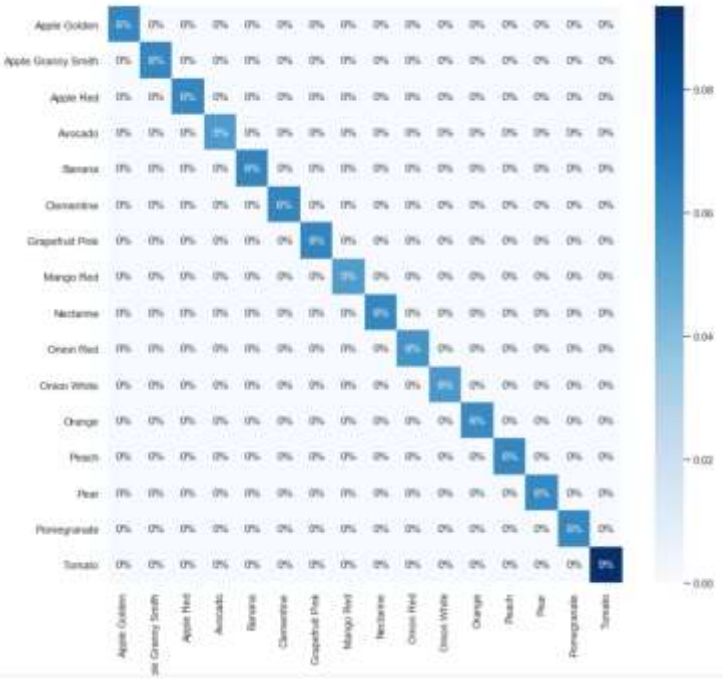
Assessing the CNN Classifier with Traditional Interpretation Methods

```
train_score = cnn_fruits_mdl.evaluate(X_train, ohe.transform(y_train),\
                                      verbose=0)
test_score = cnn_fruits_mdl.evaluate(X_test, ohe.transform(y_test),\
                                     verbose=0)
val_score = cnn_fruits_mdl.evaluate(X_val, ohe.transform(y_val),\
                                    verbose=0)
print('Train accuracy:\t{:.1%}'.format(train_score[1]))
print('Test accuracy:\t{:.1%}'.format(test_score[1]))
print('Val accuracy:\t{:.1%}'.format(val_score[1]))
```

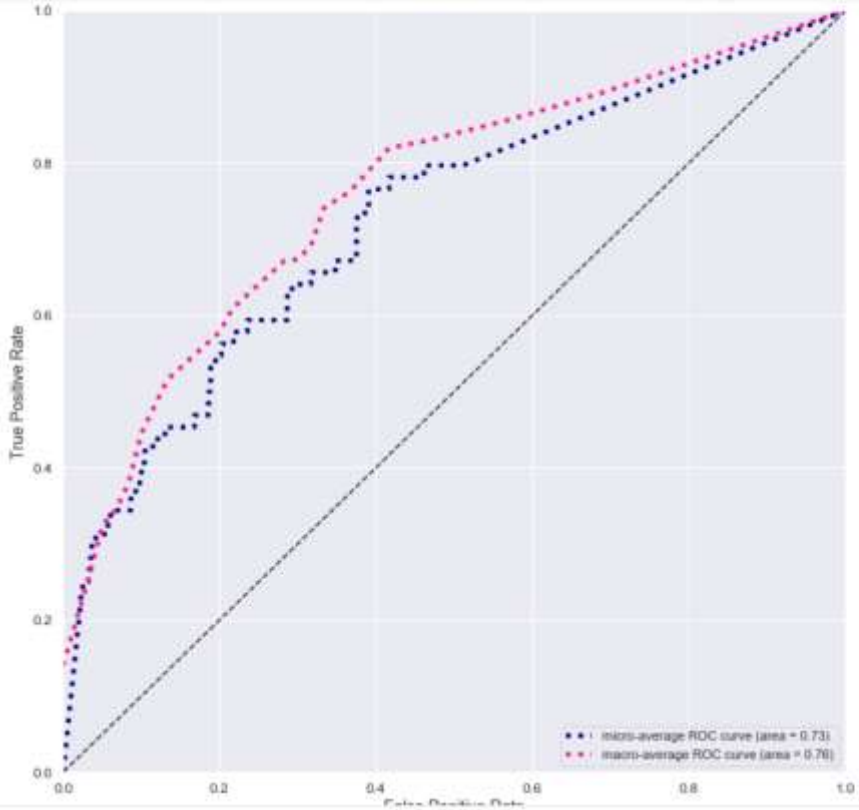
```
Train accuracy:  100.0%
Test accuracy:   99.9%
Val accuracy:    31.2%
```

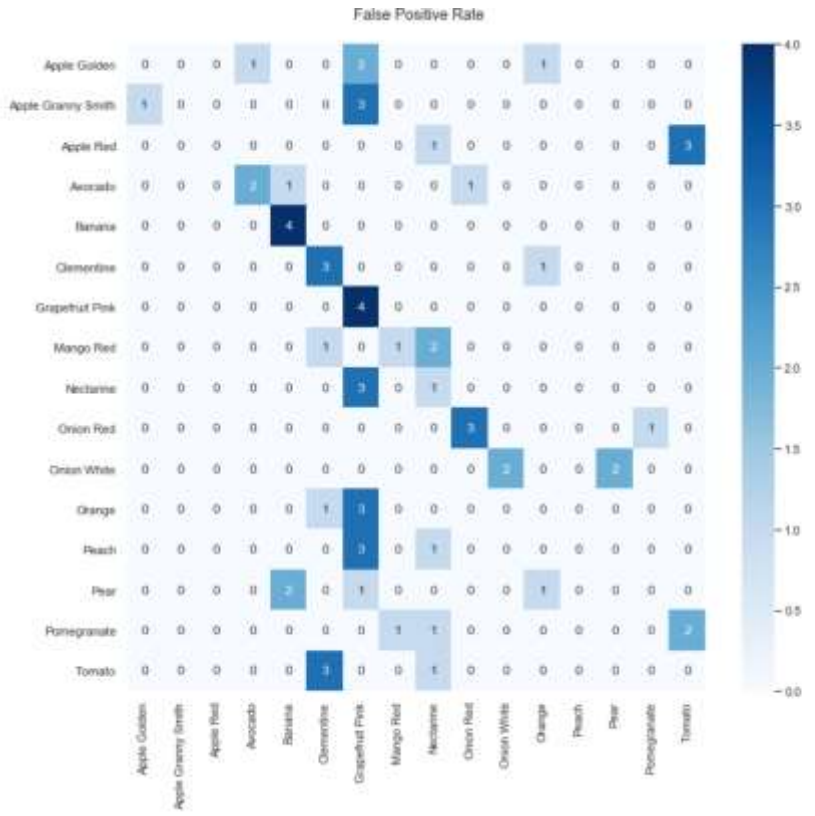
```
orig_plt_params = plt.rcParams
sns.set()
y_test_prod, y_test_prob =\
            mldatasets.evaluate_multiclass_mdl(cnn_fruits_mdl, X_test, y_test,\
                                fruits_l, ohe, plot_roc=False)
```


True Positive Rate
False Positive Rate
micro-average ROC curve (area = 1.00)
macro-average ROC curve (area = 1.00)


Apple Golden
Apple Granny Smith
Apple Red
Avocado
Banana
Clementine
Grapefruit Pink
Mango Red
Nectarine
Onion Red
Onion White
Orange
Peach
Pear
Pomegranate
Tomato

```
y_val_pred, y_val_prob =\
          mldatasets.evaluate_multiclass_mdl(cnn_fruits_mdl, X_val, y_val, fruits_l,\
                                             ohe, plot_roc=True, plot_roc_class=False,\
                                             pct_matrix=False)
```

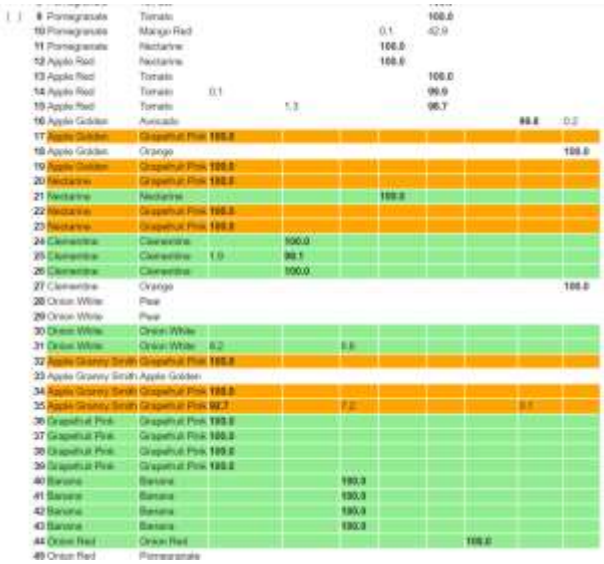




Determining what Misclassifications to Focus On

```
preds_df = pd.DataFrame({'y_true':y_val[:,0], 'y_pred':y_val_pred})

probs_df = pd.DataFrame(y_val_prob*100).round(1)
probs_df.loc['Total']= probs_df.sum().round(1)
probs_df.columns = fruits_l
probs_df = probs_df.sort_values('Total', axis=1, ascending=False)
probs_df.drop(['Total'], axis=0, inplace=True)
probs_final_df = probs_df.iloc[:,0:8]

preds_probs_df = pd.concat([preds_df, probs_final_df], axis=1)
pd.set_option('precision', 1)
preds_probs_df.style.apply(lambda x: ['background: lightgreen' if (x[0] == x[1])\
                    else '' for i in x], axis=1).\
    apply(lambda x: ['background: orange' if (x[0] != x[1] and x[1] == 'Grapefruit Pink')\
                    else '' for i in x], axis=1).\
    apply(lambda x: ['background: yellow' if (x[0] != x[1] and x[0] == 'Avocado')\
                    else '' for i in x], axis=1).\
    apply(lambda x: ['font-weight: bold' if isinstance(i, float) and i >= 50\
                    else '' for i in x], axis=1).\
    apply(lambda x: ['color:transparent' if i == 0.0\
                    else '' for i in x], axis=1)
```

	y_true	y_pred	Grapefruit Pink	Clementine	Banana	Nectarine	Tomato	Onion Red	Avocado	Orange
0	Pear	Banana			100.0					
1	Pear	Banana			100.0					
2	Pear	Orange	5.2							94.8
3	Pear	Grapefruit Pink	96.5		3.5					
4	Avocado	Onion Red						100.0		
5	Avocado	Avocado						0.1	99.9	
6	Avocado	Banana			100.0					
7	Avocado	Avocado							100.0	
8	Pomegranate	Tomato					100.0			
9	Pomegranate	Tomato					100.0			
10	Pomegranate	Mango Red				0.1	42.9			
11	Pomegranate	Nectarine				100.0				
12	Apple Red	Nectarine				100.0				



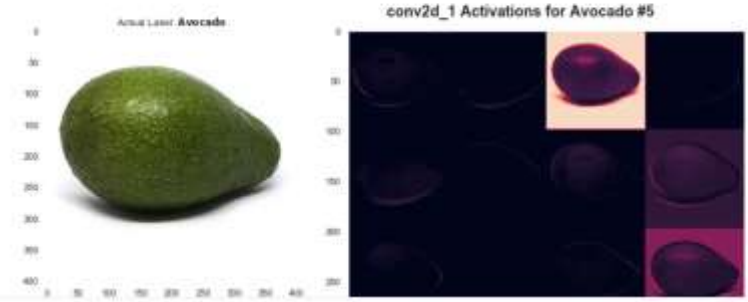
31	Onion White	Onion White	8.2		0.0					
32	Apple Granny Smith	Grapefruit Pink	100.0							
33	Apple Granny Smith	Apple Golden								
34	Apple Granny Smith	Grapefruit Pink	100.0							
35	Apple Granny Smith	Grapefruit Pink	92.7		7.2				0.1	
36	Grapefruit Pink	Grapefruit Pink	100.0							
37	Grapefruit Pink	Grapefruit Pink	100.0							
38	Grapefruit Pink	Grapefruit Pink	100.0							
39	Grapefruit Pink	Grapefruit Pink	100.0							
40	Banana	Banana			100.0					
41	Banana	Banana			100.0					
42	Banana	Banana			100.0					
43	Banana	Banana			100.0					
44	Onion Red	Onion Red						100.0		
45	Onion Red	Pomegranate								
46	Onion Red	Onion Red						100.0		
47	Onion Red	Onion Red						100.0		
48	Tomato	Clementine		100.0						
49	Tomato	Nectarine				100.0				
50	Tomato	Clementine		98.3			1.7			
51	Tomato	Clementine		100.0						
52	Mango Red	Mango Red					0.4			
53	Mango Red	Nectarine				97.1				
54	Mango Red	Clementine		100.0						
55	Mango Red	Nectarine				100.0				
56	Orange	Clementine	38.7	61.3						
57	Orange	Grapefruit Pink	100.0							
58	Orange	Grapefruit Pink	51.8	48.2						
59	Orange	Grapefruit Pink	91.6	8.4						
60	Peach	Grapefruit Pink	100.0							
61	Peach	Grapefruit Pink	100.0							
62	Peach	Nectarine		0.1		99.9				
63	Peach	Grapefruit Pink	87.7				11.9			

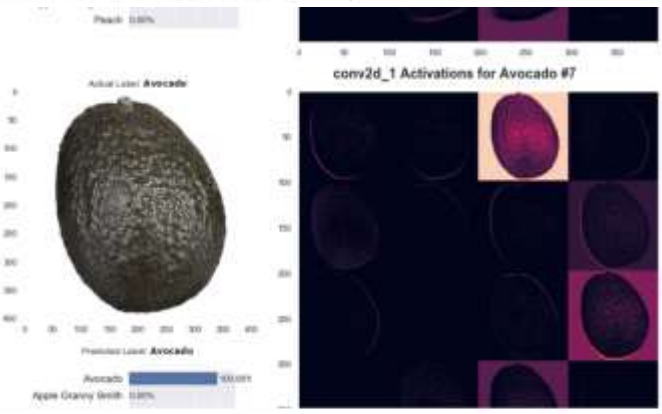
Visualizing the Learning Process with Activation-Based Methods

Intermediate Activations

```
target_layers = ['conv2d_1', 'conv2d_2', 'conv2d_3', 'conv2d_4']
explainer = ExtractActivations()
```

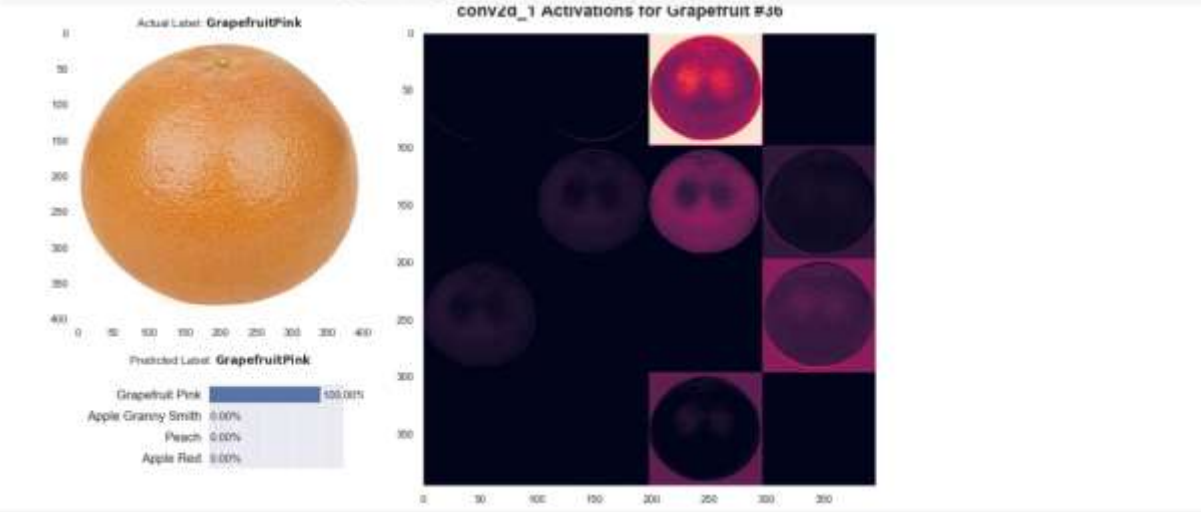
```
for target_layer in target_layers:
    for idx in avocado_TP_idxs:
        orig_img = X_val_orig[idx]
        viz_img = explainer.explain((np.array([X_val[idx]]), None),\
                                    cnn_fruits_mdl, target_layer)
        y_true = y_val[idx,0]
        y_pred = y_val_pred[idx]
        probs_s = probs_df.loc[idx]
        title = '{} Activations for Avocado #{}'.format(target_layer, idx)
        mldatasets.compare_img_pred_viz(orig_img, viz_img, y_true,\
                                        y_pred, probs_s, title=title)
```

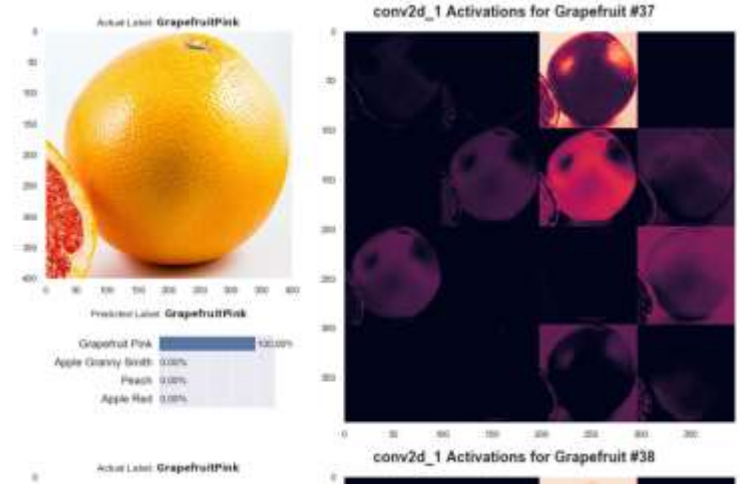





```
for target_layer in target_layers:
    for idx in grapefruit_TP_idxs:
        orig_img = X_val_orig[idx]
        viz_img = explainer.explain((np.array([X_val[idx]]), None),\
                                    cnn_fruits_mdl, target_layer)
        viz_img = np.uint8(cm.jet(viz_img)[..., :3] * 255)
        y_true = y_val[idx,0]
        y_pred = y_val_pred[idx]
        probs_s = probs_df.loc[idx]
        title = '{} Activations for Grapefruit #{}'.format(target_layer, idx)
```

```
[ ]  for target_layer in target_layers:
         for idx in grapefruit_TP_idxs:
             orig_img = X_val_orig[idx]
             viz_img = explainer.explain((np.array([X_val[idx]]), None),\
                                         cnn_fruits_mfl, target_layer)
             #viz_img = np.uint8(cm.jet(viz_img)[..., :3] * 255)
             y_true = y_val[idx,0]
             y_pred = y_val_pred[idx]
             probs_s = probs_df.loc[idx]
             title = '{} Activations for Grapefruit #{}'.format(target_layer, idx)
             mldatasets.compare_img_pred_viz(orig_img, viz_img, y_true,\
                                             y_pred, probs_s, title=title)
```



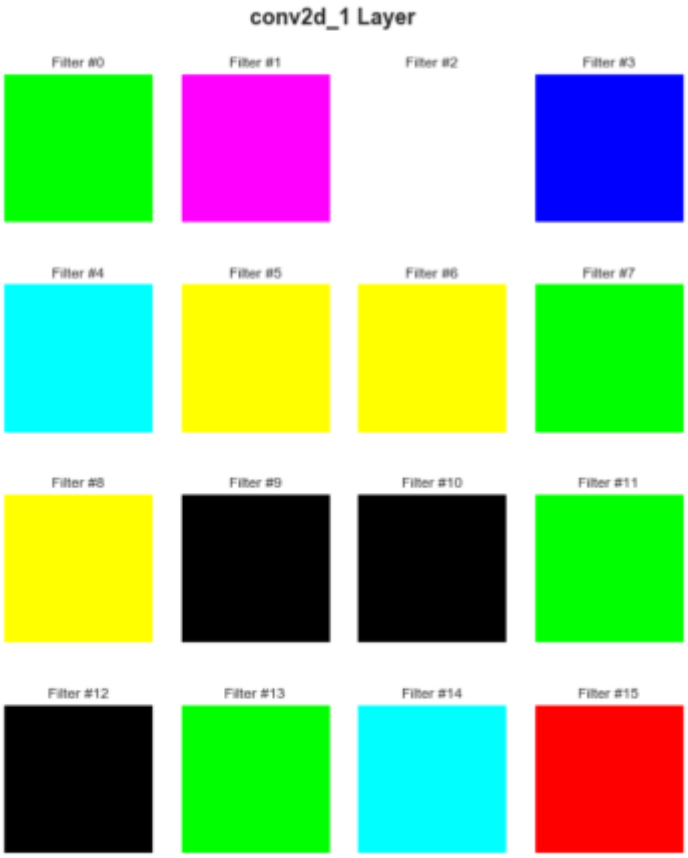


Activation Maximization

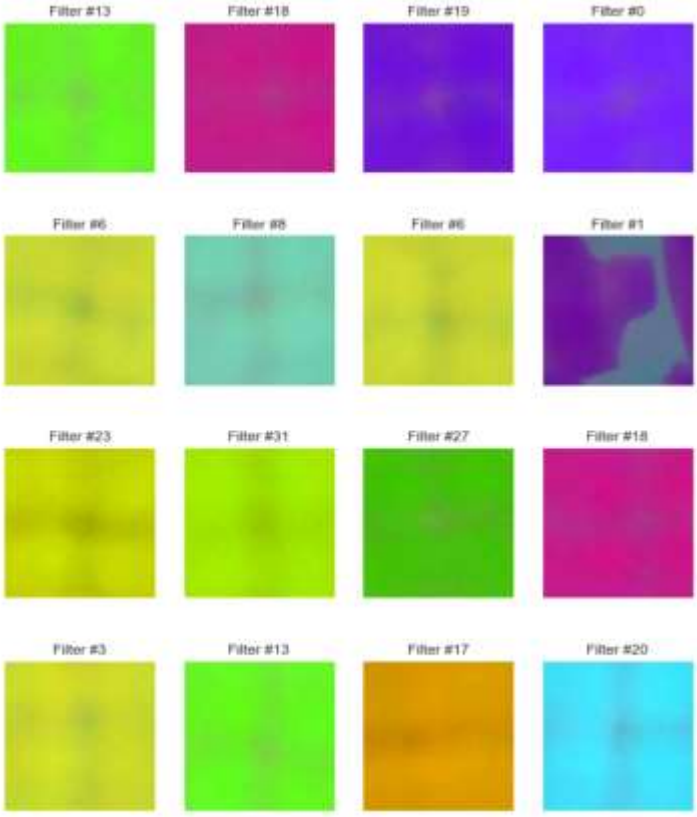
```
def model_modifier(mdl):
    global target_layer
    target = mdl.get_layer(name=target_layer)
    new_mdl = tf.keras.Model(inputs=mdl.inputs,\
                             outputs=target.output)
    new_mdl.layers[-1].activation = tf.keras.activations.linear
    return new_mdl

def loss(output):
    global filter_num
    return output[..., filter_num]
```

```
[ ] #How many filters to plot Activation Maximization for in each layer
    num_filters = 16

    #Compute size (width or height) of image size based on num_filters
    gridsize = math.ceil(math.sqrt(num_filters))

    #Iterate each target layer and..
    for target_layer in target_layers:

        #Randomly select index of filters from total amount in layer
        for layer in cnn_fruits_mdl.layers:
            if layer.name == target_layer:
                total_filters = layer.filters
        if total_filters == num_filters or total_filters < num_filters:
            filter_num_l = [*range(num_filters)]
        else:
            filter_num_l = list(np.random.choice([*range(total_filters)],\
                                                 num_filters))

        #Compute and Plot Activation Maximization for each random filter
        fig = plt.figure(figsize=(10,10))
        for f, filter_num in zip([*range(len(filter_num_l))], filter_num_l):
            plt.subplot(gridsize, gridsize, f+1)
            plt.title('Filter #{}'.format(filter_num), fontsize=12)
            activation_maximization = ActivationMaximization(cnn_fruits_mdl,\
```

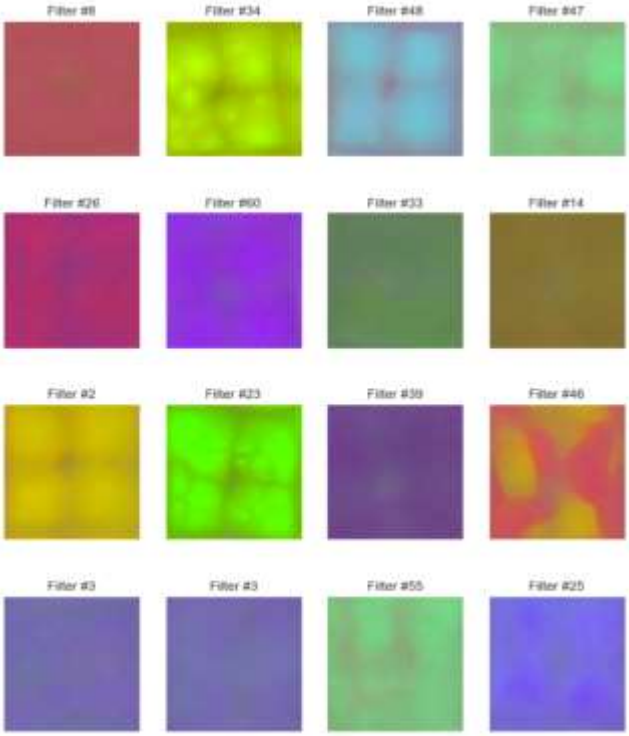
[ ]



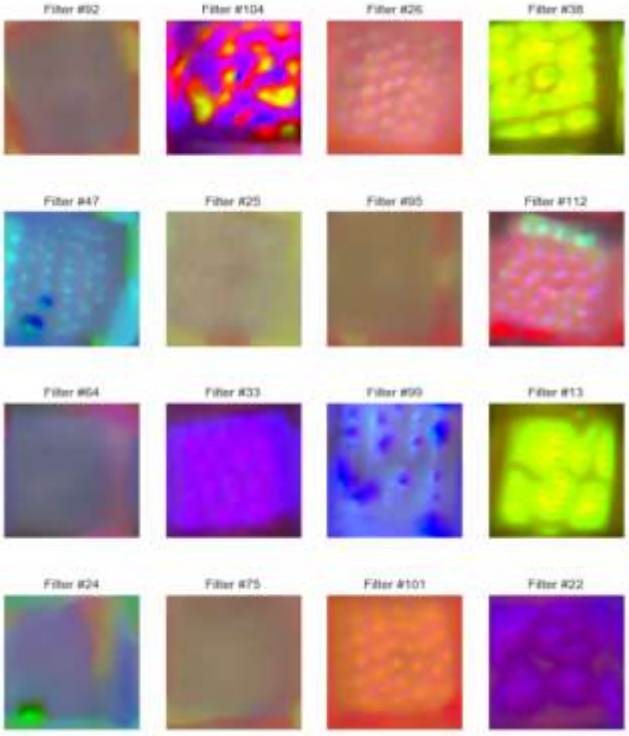
conv2d_2 Layer



conv2d_3 Layer



conv2d_4 Layer



Evaluating Misclassifications with Gradient-Based Attribution Methods

```
[ ] idxs = avocado_FN_idxs + grapefruit_FP_idxs
    X_misclass = X_val[idxs]
    print(X_misclass.shape)
```

```
(17, 100, 100, 3)
```

```
[ ] enc = preprocessing.OrdinalEncoder()
    enc.fit(y_train)
    y_val_pred_exp = np.expand_dims(np.array(y_val_pred),axis=1)
    y_val_pred_enc = enc.transform(y_val_pred_exp)
    labels_1 = y_val_pred_enc[idxs].squeeze().\
                                        astype(int).tolist()
    print(labels_1)
```

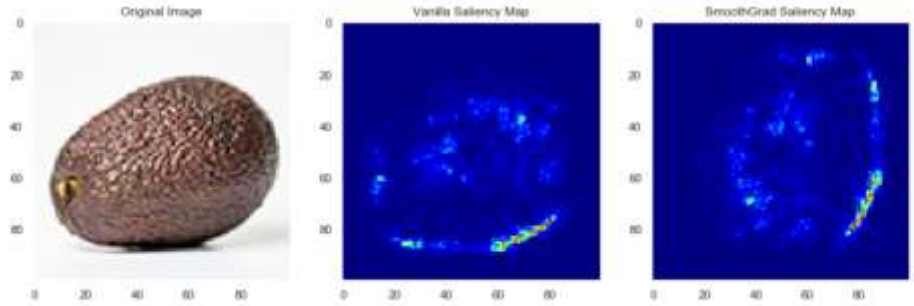
```
[9, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6]
```

```
[ ] def model_modifier(mdl):
        mdl.layers[-1].activation = tf.keras.activations.linear
        return mdl

    def loss(output):
        global labels_1
        pos_1 = [*range(len(labels_1))]
        output_1 = []
        for p, l in zip(pos_1, labels_1):
            output_1.append(output[p][l])
        return tuple(output_1)
```

Saliency Maps

```
[ ] saliency = Saliency(cnn_fruits_mdl,
                        model_modifier=model_modifier,
                        clone=True)
```

```
[ ] plt.subplots(figsize=(15,5))
    plt.subplot(1, 3, 1)
    plt.imshow(X_misclass[0])
    plt.grid(b=None)
    plt.title("Original Image")
    plt.subplot(1, 3, 2)
    plt.imshow(saliency_maps[0], cmap='jet')
    plt.grid(b=None)
    plt.title("Vanilla Saliency Map")
    plt.subplot(1, 3, 3)
    plt.imshow(smoothgrad_saliency_maps[0], cmap='jet')
    plt.grid(b=None)
    plt.title("SmoothGrad Saliency Map")
    plt.show()
```



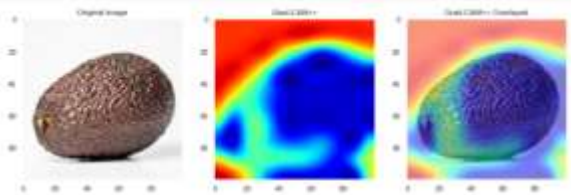

```
[ ] print(y_val_pred[idxs[0]])
```

Grad-CAM

Creating GradCam++ Maps

```
[ ] gradcam = GradcamPlusPlus(cnn_fruits_mdl,\
                              model_modifier,\
                              clone=True)

    gradcam_maps = gradcam(loss, X_misclass,\
                           penultimate_layer=-1)
    gradcam_maps = normalize(gradcam_maps)
```

```
[ ] plt.subplots(figsize=(15,5))
    plt.subplot(1, 3, 1)
    plt.imshow(X_misclass[0])
    plt.grid(b=None)
    plt.title("Original Image")
    plt.subplot(1, 3, 2)
    heatmap = np.uint8(cm.jet(gradcam_maps[0])[..., :3] * 255)
    plt.imshow(heatmap)
    plt.grid(b=None)
    plt.title("Grad-CAM++")
    plt.subplot(1, 3, 3)
    plt.imshow(X_misclass[0])
    plt.imshow(heatmap, alpha=0.5)
    plt.grid(b=None)
    plt.title("Grad-CAM++ Overlayed")
    plt.show()
```

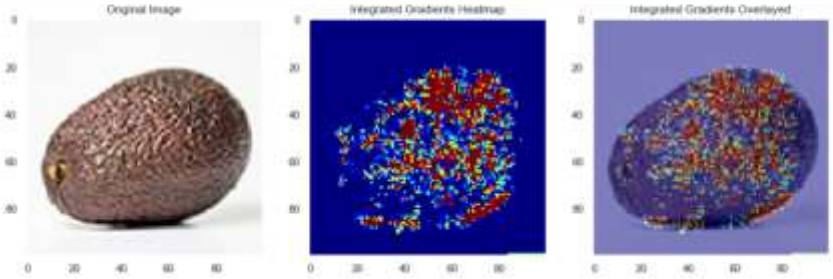


Integrated Gradients

```
[ ] explainer = IntegratedGradients()

    ig_maps = []
    for i in range(len(labels_)):
        img = (X_misclass[i]), None)
        label = labels_[i]
        ig_map = explainer.explain(img, cnn_fruits_mdl,\
                                   label, n_steps=20)
        ig_maps.append(ig_map)
```

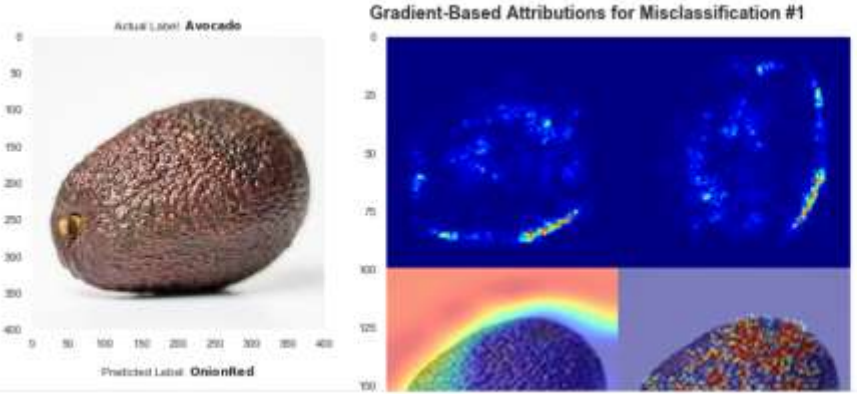
```
[ ] plt.subplots(figsize=(15,5))
    plt.subplot(1, 3, 1)
    plt.imshow(X_misclass[0])
    plt.grid(b=None)
    plt.title("Original Image")
```

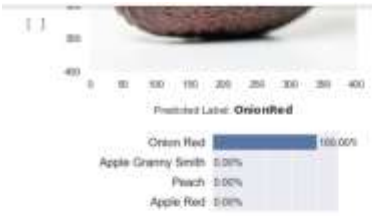
```
[ ] plt.subplots(figsize=(15,5))
    plt.subplot(1, 3, 1)
    plt.imshow(X_misclass[0])
    plt.grid(b=None)
    plt.title("Original Image")
    plt.subplot(1, 3, 2)
    heatmap = np.uint8(cm.jet(ig_maps[0])[..., :3] * 255)
    plt.imshow(heatmap)
    plt.grid(b=None)
    plt.title("Integrated Gradients Heatmap")
    plt.subplot(1, 3, 3)
    plt.imshow(X_misclass[0])
    plt.imshow(heatmap, alpha=0.5)
    plt.grid(b=None)
    plt.title("Integrated Gradients Overlayed")
    plt.show()
```

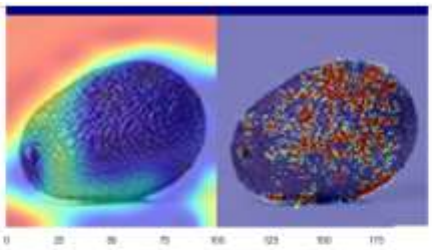


Tying it All Together

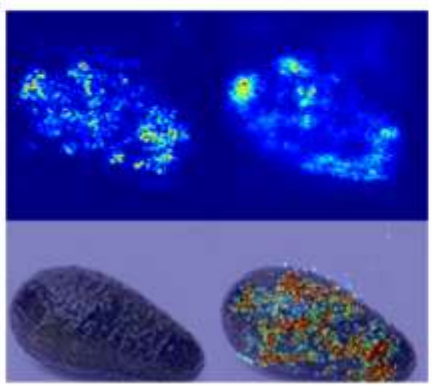
```
[ ] for pos, idx in zip([*range(len(idxs))], idxs):
        orig_img = X_val_orig[idx]
        map1 = np.uint8(cm.jet(saliency_maps[pos])[..., :3] * 255)
        map2 = np.uint8(cm.jet(smoothgrad_saliency_maps[pos])[..., :3] * 255)
        map3 = mldatasets.heatmap_overlay(X_misclass[pos], gradcam_maps[pos])
        map4 = mldatasets.heatmap_overlay(X_misclass[pos], ig_maps[pos])
        viz_img = cv2.vconcat([
                cv2.hconcat([map1, map2]),
                cv2.hconcat([map3, map4])
            ])
        y_true = y_val[idx,0]
        y_pred = y_val_pred[idx]
        probs_s = probs_df.loc[idx]
        title = 'Gradient-Based Attributions for Misclassification #{}'.format(pos+1)
        mldatasets.compare_img_pred_viz(orig_img, viz_img, y_true,\
                                        y_pred, probs_s, title=title)
```

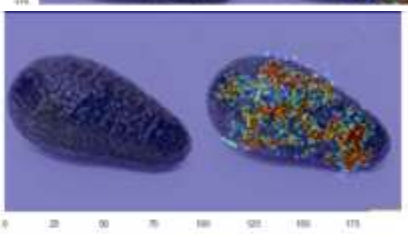




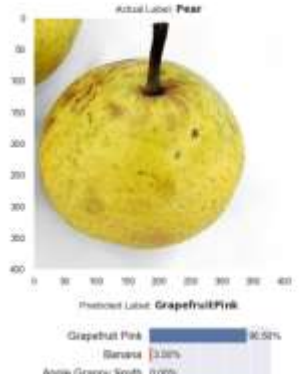


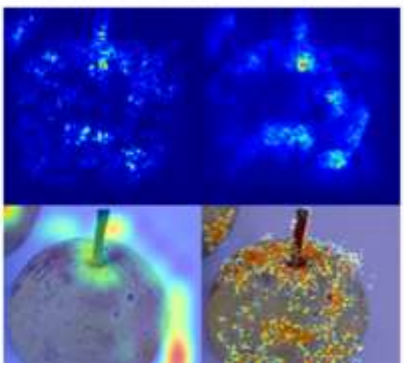
Gradient-Based Attributions for Misclassification #2



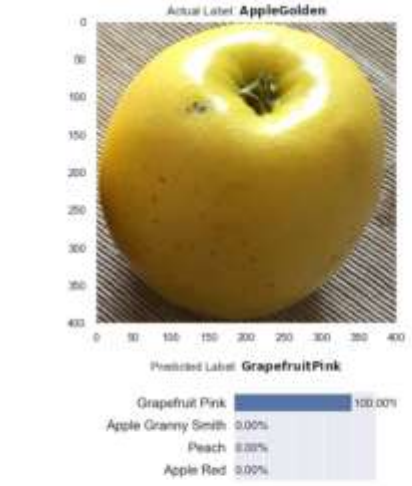


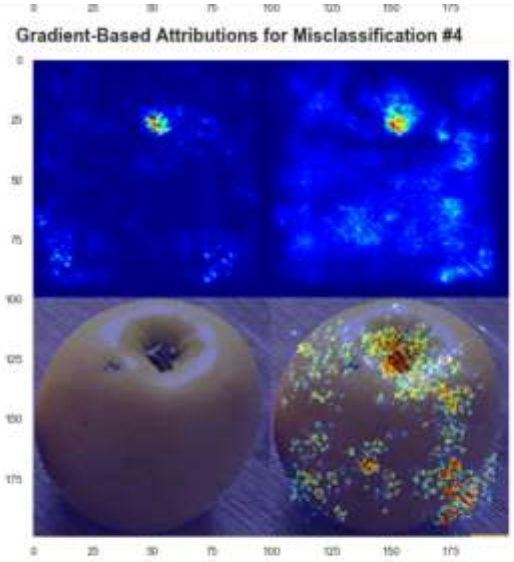
Gradient-Based Attributions for Misclassification #3





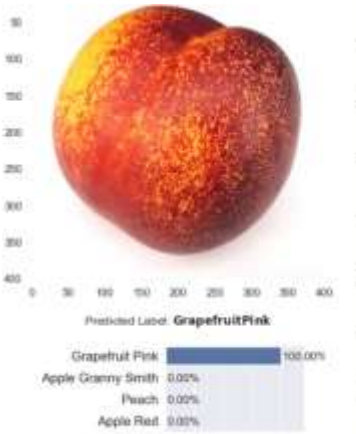
[ ]

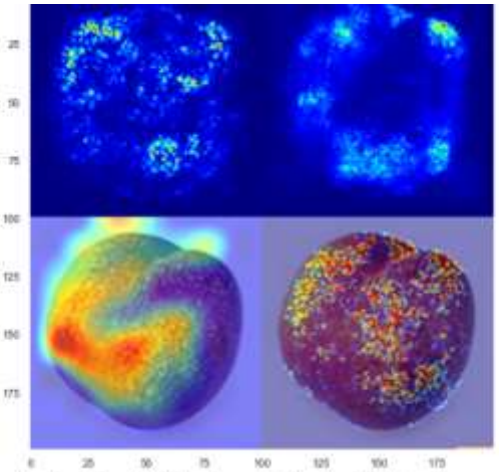

Actual Label: AppleGolden
Predicted Label: GrapefruitPink
Grapefruit Pink 100.00%
Apple Granny Smith 0.00%
Peach 0.00%
Apple Red 0.00%


Gradient-Based Attributions for Misclassification #4

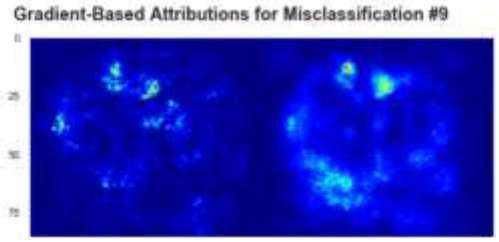

Actual Label: AppleGolden

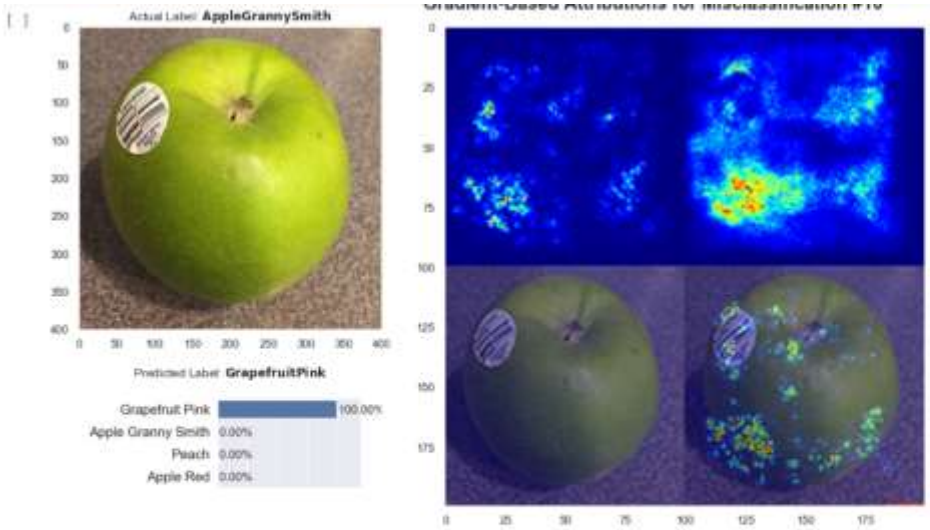

Gradient-Based Attributions for Misclassification #5

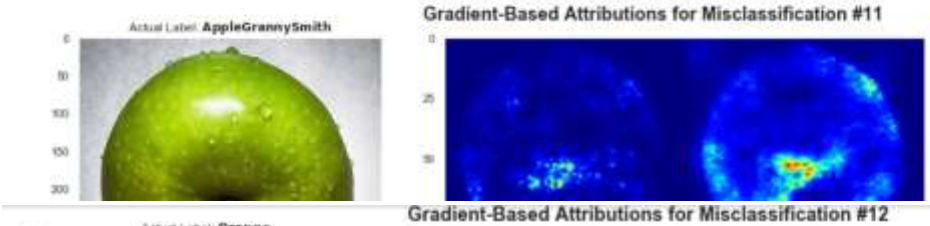

Predicted Label: GrapefruitPink
Grapefruit Pink 100.00%
Apple Granny Smith 0.00%
Peach 0.00%
Apple Red 0.00%

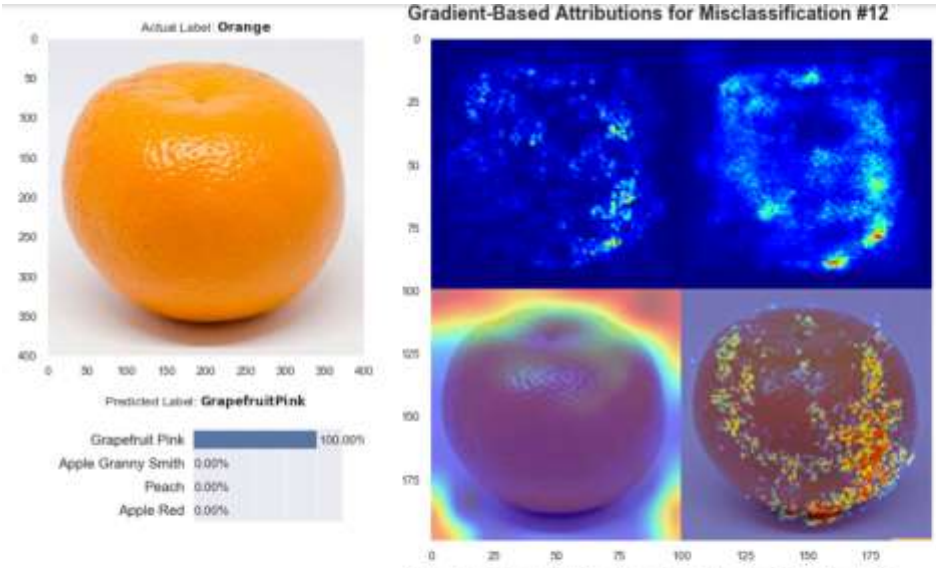


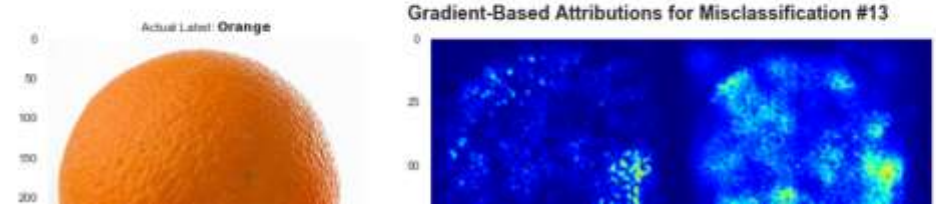

Actual Label: AppleGrannySmith

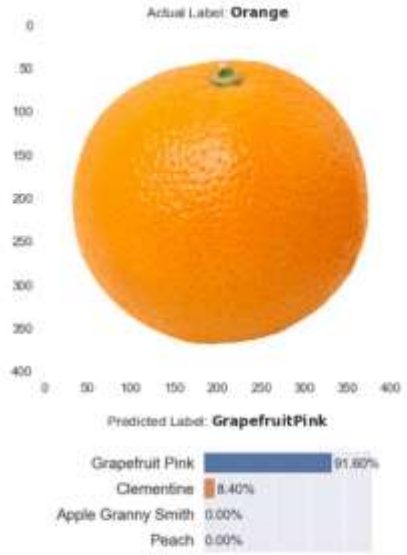

Gradient-Based Attributions for Misclassification #9

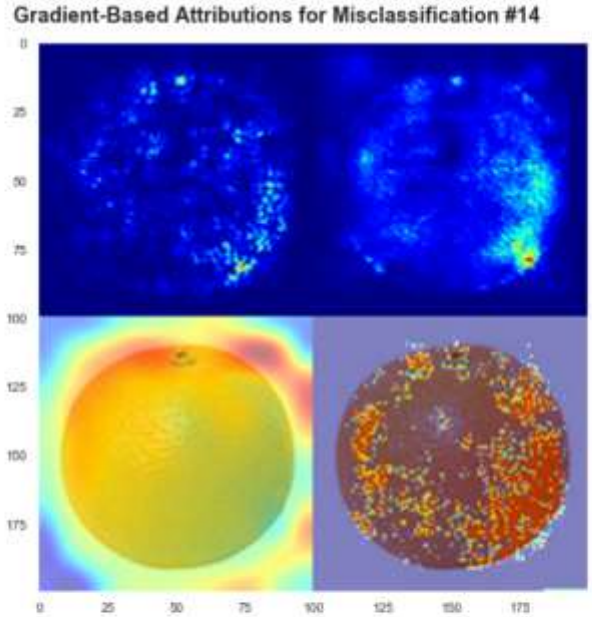

Actual Label: AppleGrannySmith
Predicted Label: GrapefruitPink
Grapefruit Pink 100.00%
Apple Granny Smith 0.00%
Peach 0.00%
Apple Red 0.00%


Actual Label: AppleGrannySmith
Gradient-Based Attributions for Misclassification #11

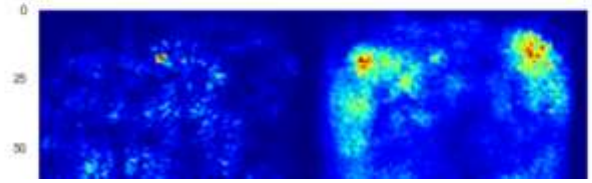

Actual Label: Orange
Predicted Label: GrapefruitPink
Grapefruit Pink 100.00%
Apple Granny Smith 0.00%
Peach 0.00%
Apple Red 0.00%
Gradient-Based Attributions for Misclassification #12


Actual Label: Orange
Gradient-Based Attributions for Misclassification #13

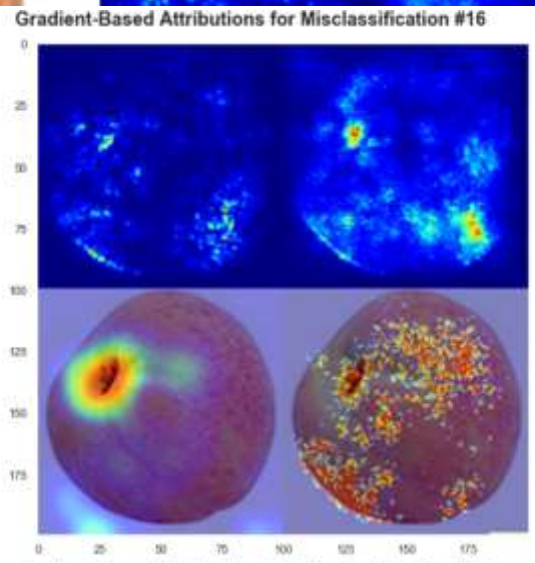

Actual Label: Orange
Predicted Label: GrapefruitPink
Grapefruit Pink 91.60%
Clementine 8.40%
Apple Granny Smith 0.00%
Peach 0.00%


Gradient-Based Attributions for Misclassification #14

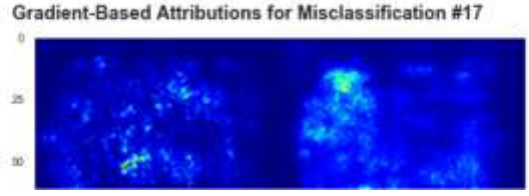

Actual Label: Peach

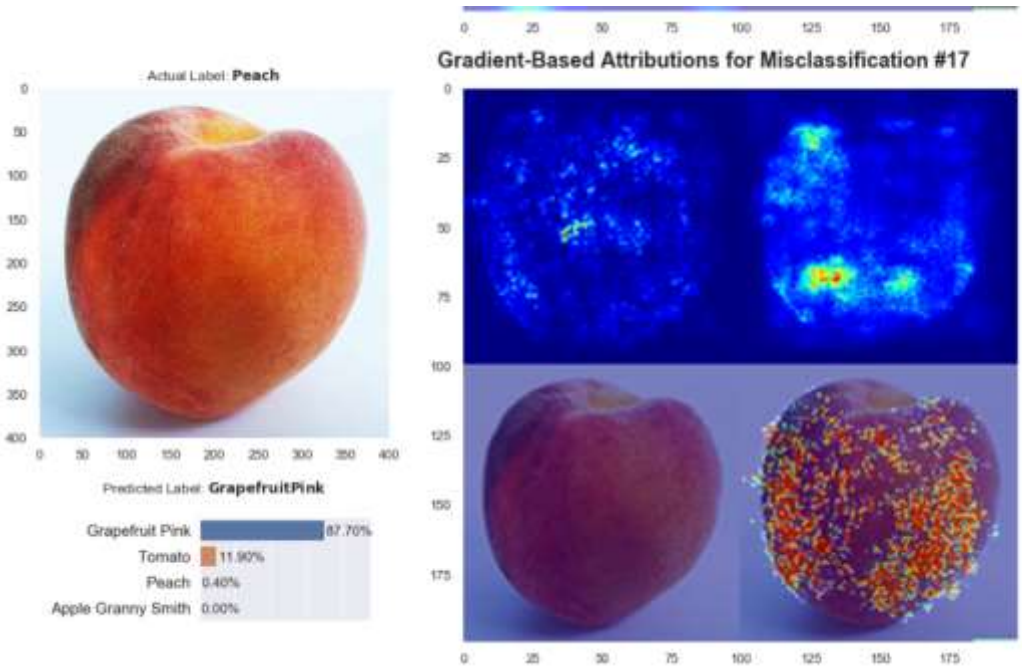

Gradient-Based Attributions for Misclassification #15


Actual Label: Peach
Predicted Label: GrapefruitPink
Grapefruit Pink 100.00%
Apple Granny Smith 0.00%
Peach 0.00%
Apple Red 0.00%


Gradient-Based Attributions for Misclassification #16


Actual Label: Peach


Gradient-Based Attributions for Misclassification #17



Part 2:

```
import math
import os
import machine_learning_datasets as mldatasets
import numpy as np
import pandas as pd
from sklearn import preprocessing, metrics
import tensorflow as tf
tf.compat.v1.disable_v2_behavior()
from tensorflow import keras
import matplotlib.pyplot as plt
from matplotlib import cm
import seaborn as sns
import cv2
from keras.utils.data_utils import get_file
#PART 2 only
from skimage.segmentation import mark_boundaries
from tf_explain.core.occlusion_sensitivity import OcclusionSensitivity
import lime
from lime import lime_image
from alibi.explainers import CEM
import shap
```

```
print('TF version:\t\t', tf.__version__)
print('Eager exec enabled:\t', tf.executing_eagerly())
```

```
TF version:             2.2.0
Eager exec enabled:     False
```

```
X_train, X_test, X_val, X_val_orig, y_train, y_test, y_val, y_val_orig =\
                                    mldatasets.load('fruits-360', prepare=True)
```

```
https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python/raw/master/datasets/fruits-360_abbrev.zip downloaded to /Users/serg/EXTEND/PROJECTS/
/Users/serg/EXTEND/PROJECTS/InterpretableMLBook/programming/Chapter8/data/fruits-360_abbrev.zip uncompressed to /Users/serg/EXTEND/PROJECTS/InterpretableMLBook/pr
10633 dataset files found in /Users/serg/EXTEND/PROJECTS/InterpretableMLBook/programming/Chapter8/data/fruits-360_abbrev Folder
```

+ Code + Text

```
X_train = X_train.astype('float32')/255
X_test = X_test.astype('float32')/255
X_val = X_val.astype('float32')/255
```

+ Code + Text

```
ohe = preprocessing.OneHotEncoder(sparse=False)
ohe.fit(y_train)
fruits_l = ohe.categories_[0].tolist()
print(fruits_l)
```

```
['Apple Golden', 'Apple Granny Smith', 'Apple Red', 'Avocado', 'Banana', 'Clementine', 'Grapefruit Pink', 'Mango Red', 'Nectarine', 'Onion Red', 'Onion White',
```

```
rand = 0
os.environ['PYTHONHASHSEED']=str(rand)
np.random.seed(rand)
tf.random.set_seed(rand)
```

```
model_path = get_file('CNN_fruits_final.hdf5',\
        'https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python/blob/master/models/CNN_fruits_final.hdf5?raw=true')
cnn_fruits_mdl = keras.models.load_model(model_path)
cnn_fruits_mdl.summary()
```

```
Model: "CNN_fruits"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 99, 99, 16)        208
_________________________________________________________________
maxpool2d_1 (MaxPooling2D)   (None, 49, 49, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 48, 48, 32)        2080
_________________________________________________________________
maxpool2d_2 (MaxPooling2D)   (None, 24, 24, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 23, 23, 64)        8256
_________________________________________________________________
maxpool2d_3 (MaxPooling2D)   (None, 11, 11, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 10, 10, 128)       32896
_________________________________________________________________
maxpool2d_4 (MaxPooling2D)   (None, 5, 5, 128)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 5, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 3200)              0
_________________________________________________________________
dense_1 (Dense)              (None, 150)               480150
_________________________________________________________________
dropout_2 (Dropout)          (None, 150)               0
_________________________________________________________________
dense_2 (Dense)              (None, 16)                2416
=================================================================
Total params: 526,006
```

```
y_val_pred, y_val_prob =\
            mldatasets.evaluate_multiclass_mdl(cnn_fruits_mdl, X_val, y_val, fruits_l,\
                                ohe, plot_roc=False, plot_conf_matrix=False, pct_matrix=False)
```

```
                    precision    recall  f1-score   support

      Apple Golden      0.000     0.000     0.000         4
Apple Granny Smith      0.000     0.000     0.000         4
         Apple Red      0.000     0.000     0.000         4
           Avocado      0.667     0.500     0.571         4
            Banana      0.571     1.000     0.727         4
        Clementine      0.375     0.750     0.500         4
   Grapefruit Pink      0.211     1.000     0.348         4
         Mango Red      0.500     0.250     0.333         4
         Nectarine      0.143     0.250     0.182         4
         Onion Red      0.750     0.750     0.750         4
       Onion White      1.000     0.500     0.667         4
            Orange      0.000     0.000     0.000         4
             Peach      0.000     0.000     0.000         4
              Pear      0.000     0.000     0.000         4
       Pomegranate      0.000     0.000     0.000         4
            Tomato      0.000     0.000     0.000         4

          accuracy                          0.312        64
         macro avg      0.264     0.312     0.255        64
      weighted avg      0.264     0.312     0.255        64
```

```
enc = preprocessing.OrdinalEncoder()
enc.fit(y_train)
y_val_pred_exp = np.expand_dims(np.array(y_val_pred),axis=1)
y_val_pred_enc = enc.transform(y_val_pred_exp)
```

```
preds_df = pd.DataFrame({'y_true':y_val[:,0], 'y_pred':y_val_pred})

probs_df = pd.DataFrame(y_val_prob*100).round(1)
probs_df.loc['Total']= probs_df.sum().round(1)
probs_df.columns = fruits_l
probs_df = probs_df.sort_values('Total', axis=1, ascending=False)
probs_df.drop(['Total'], axis=0, inplace=True)
```

```
avocado_FN_idxs = preds_df[(preds_df['y_true'] != preds_df['y_pred']) &\
                           (preds_df['y_true'] == 'Avocado')].index.to_list()
avocado_TP_idxs = preds_df[(preds_df['y_true'] == preds_df['y_pred']) &\
                           (preds_df['y_true'] == 'Avocado')].index.to_list()
grapefruit_FP_idxs = preds_df[(preds_df['y_true'] != preds_df['y_pred']) &\
                              (preds_df['y_pred'] == 'Grapefruit Pink')].index.to_list()
grapefruit_TP_idxs = preds_df[(preds_df['y_true'] == preds_df['y_pred']) &\
                              (preds_df['y_pred'] == 'Grapefruit Pink')].index.to_list()
```

```
idxs = avocado_TP_idxs + grapefruit_TP_idxs
X_tp = X_val[idxs]
print(X_tp.shape)
```

```
(6, 100, 100, 3)
```

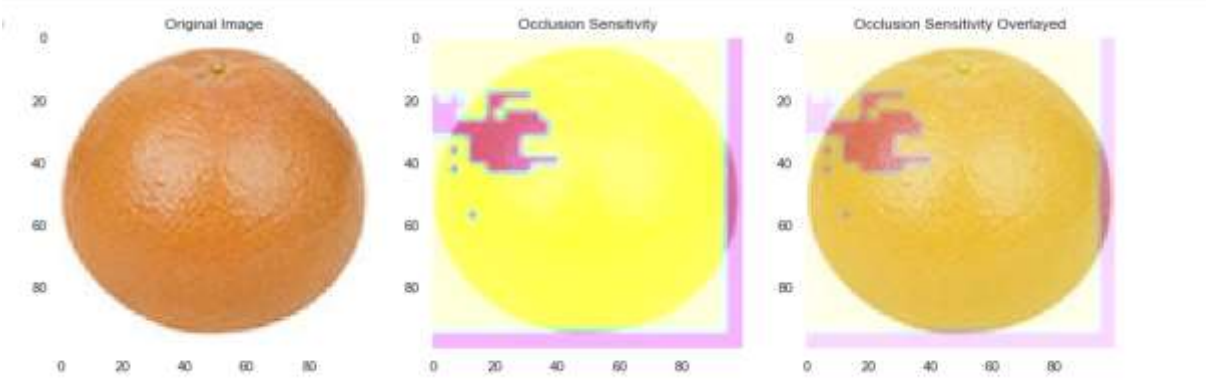
```
labels_l = y_val_pred_enc[idxs].squeeze().\
                                        astype(int).tolist()
print(labels_l)
```

```
[3, 3, 6, 6, 6, 6]
```

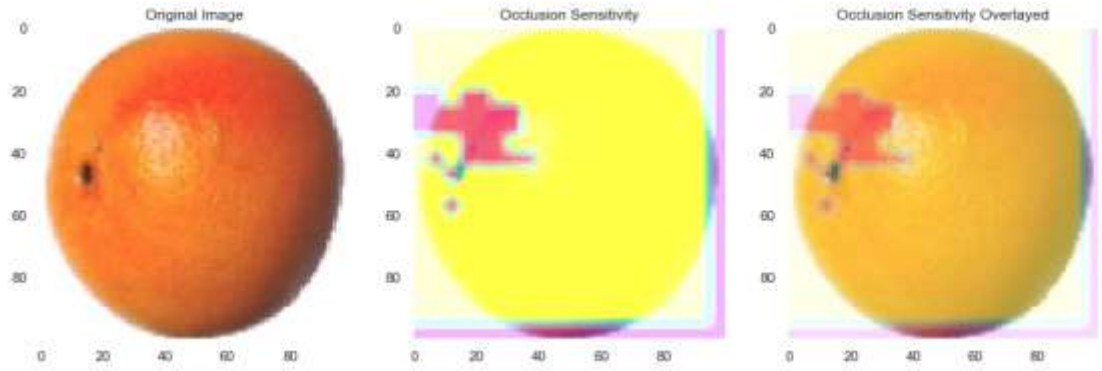
```
explainer = OcclusionSensitivity()

os_maps = []
for i in range(len(labels_l)):
    img = ([X_tp[i]], None)
    label = labels_l[i]
    os_map = explainer.explain(img, cnn_fruits_mdl,\
                               label, 5)
    os_maps.append(os_map)
```

```
explainer = OcclusionSensitivity()

os_maps = []
for i in range(len(labels_l)):
    img = ([X_tp[i]], None)
    label = labels_l[i]
    os_map = explainer.explain(img, cnn_fruits_mdl,\
                               label, 3)
    os_maps.append(os_map)
```

```
plt.subplots(figsize=(15,5))
plt.subplot(1, 3, 1)
plt.imshow(X_tp[2])
plt.grid(b=None)
plt.title("Original Image")
plt.subplot(1, 3, 2)
plt.imshow(os_maps[2])
plt.grid(b=None)
plt.title("Occlusion Sensitivity")
plt.subplot(1, 3, 3)
plt.imshow(X_tp[2])
plt.imshow(os_maps[2], alpha=0.5)
plt.grid(b=None)
plt.title("Occlusion Sensitivity Overlayed")
plt.show()
```



```
idx = np.random.choice(np.where(y_train[:,0] == 'Grapefruit Pink')[0], 1)[0]
os_map_train = explainer.explain(([X_train[idx]], None), cnn_fruits_mdl, label, 5)
plt.subplots(figsize=(15,5))
plt.subplot(1, 3, 1)
plt.imshow(X_train[idx])
plt.grid(b=None)
plt.title("Original Image")
plt.subplot(1, 3, 2)
plt.imshow(os_map_train)
plt.grid(b=None)
plt.title("Occlusion Sensitivity")
plt.subplot(1, 3, 3)
plt.imshow(X_train[idx])
plt.imshow(os_map_train, alpha=0.5)
plt.grid(b=None)
plt.title("Occlusion Sensitivity Overlayed")
plt.show()
```

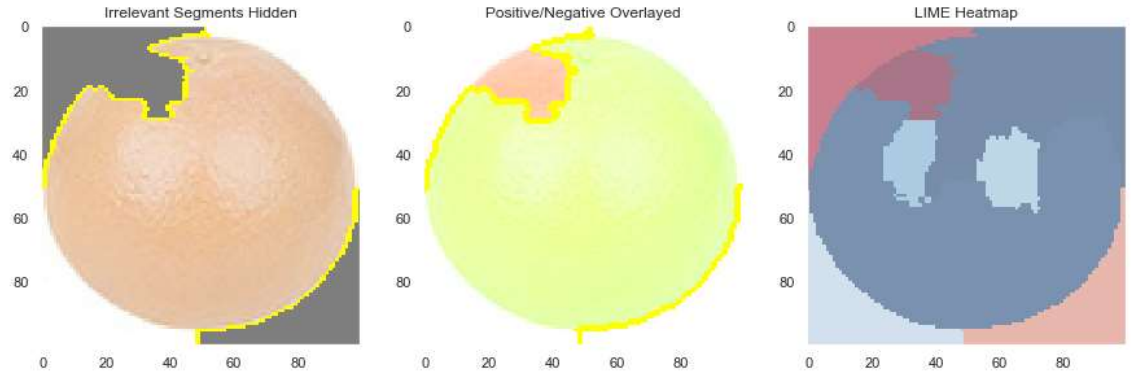



```
explainer = lime_image.LimeImageExplainer()

lime_expl = []
for i in range(len(labels_l)):
    explanation = explainer.\
                  explain_instance(X_tp[i].astype('double'),\
                          cnn_fruits_mdl.predict, top_labels=5,\
                          hide_color=0, num_samples=1000)
    lime_expl.append(explanation)
```

tracting Image and Mask from Explanation

```
#Explanation with irrelevant segments hidden
img_hide, mask_hide = lime_expl[2].\
                      get_image_and_mask(lime_expl[2].top_labels[0],\
                                 positive_only=True, num_features=10,\
                                 hide_rest=True)
img_hide = mark_boundaries(img_hide / 2 + 0.5, mask_hide)

#Explanation with all segments marked for positive/negative prediction
img_show, mask_show = lime_expl[2].\
                          get_image_and_mask(lime_expl[2].top_labels[0],\
                              positive_only=False, num_features=10)
img_show = mark_boundaries(img_show / 2 + 0.5, mask_show)

#Heatmap explanation by segment
dict_heatmap = dict(lime_expl[2].local_exp[lime_expl[2].top_labels[0]])
heatmap = np.vectorize(dict_heatmap.get)(lime_expl[2].segments)
```

```
plt.subplots(figsize=(15,5))
plt.subplot(1, 3, 1)
plt.imshow(img_hide)
plt.grid(b=None)
plt.title("Irrelevant Segments Hidden")
plt.subplot(1, 3, 2)
plt.imshow(img_show)
plt.grid(b=None)
plt.title("Positive/Negative Overlayed")
plt.subplot(1, 3, 3)
plt.imshow(heatmap, alpha=0.5, cmap='RdBu')
plt.grid(b=None)
plt.title("LIME Heatmap")
plt.show()
```



Train Autoencoder

```
input_layer = tf.keras.layers.Input(shape=(100, 100, 3))
encoder = tf.keras.layers.Conv2D(16, (3, 3), activation='relu',\
                                 padding='same')(input_layer)
encoder = tf.keras.layers.Conv2D(16, (3, 3), activation='relu',\
                                 padding='same')(encoder)
encoder = tf.keras.layers.MaxPooling2D((2, 2), padding='same')(encoder)
bottleneck = tf.keras.layers.Conv2D(1, (3, 3), activation=None,\
                                    padding='same')(encoder)
decoder = tf.keras.layers.Conv2D(16, (3, 3), activation='relu',\
                                 padding='same')(bottleneck)
decoder = tf.keras.layers.UpSampling2D((2, 2))(decoder)
decoder = tf.keras.layers.Conv2D(16, (3, 3), activation='relu',\
                                 padding='same')(decoder)
output_layer = tf.keras.layers.Conv2D(3, (3, 3), activation=None,\
                                      padding='same')(decoder)
autoencoder_mdl = tf.keras.Model(input_layer, output_layer)
autoencoder_mdl.summary()
```

```
Model: "model_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 100, 100, 3)]     0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 100, 100, 16)      448
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 100, 100, 16)      2320
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 50, 50, 16)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 50, 50, 1)         145
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 50, 50, 16)        160
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 100, 100, 16)      0
```

```
Model: "model_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 100, 100, 3)]     0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 100, 100, 16)      448
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 100, 100, 16)      2320
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 50, 50, 16)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 50, 50, 1)         145
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 50, 50, 16)        160
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 100, 100, 16)      0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 100, 100, 16)      2320
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 100, 100, 3)       435
=================================================================
Total params: 5,828
Trainable params: 5,828
Non-trainable params: 0
_________________________________________________________________
```

```
autoencoder_mdl.compile(loss='mse', optimizer='adam')
autoencoder_history = autoencoder_mdl.fit(X_train, X_train, epochs=5,\
                      batch_size=32, validation_data=(X_test, X_test),\
                      verbose=1)
```

```
Train on 7872 samples, validate on 2633 samples
Epoch 1/5
7872/7872 [==============================] - 116s 15ms/sample - loss: 0.0235 - val_loss: 0.0062
Epoch 2/5
7872/7872 [==============================] - 104s 13ms/sample - loss: 0.0057 - val_loss: 0.0053
Epoch 3/5
7872/7872 [==============================] - 102s 13ms/sample - loss: 0.0051 - val_loss: 0.0049
Epoch 4/5
7872/7872 [==============================] - 105s 13ms/sample - loss: 0.0047 - val_loss: 0.0045
Epoch 5/5
7872/7872 [==============================] - 101s 13ms/sample - loss: 0.0044 - val_loss: 0.0042
```

sess Autoencoder

```
n = 7
rand_idxs = np.random.choice([*range(len(y_test))], n)

decoded_imgs = autoencoder_mdl.predict(X_test[rand_idxs])

plt.figure(figsize=(14, 4))
for i in range(n):
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(X_test[rand_idxs[i]])
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    ax = plt.subplot(2, n, i + n + 1)
    plt.imshow(decoded_imgs[i])
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```

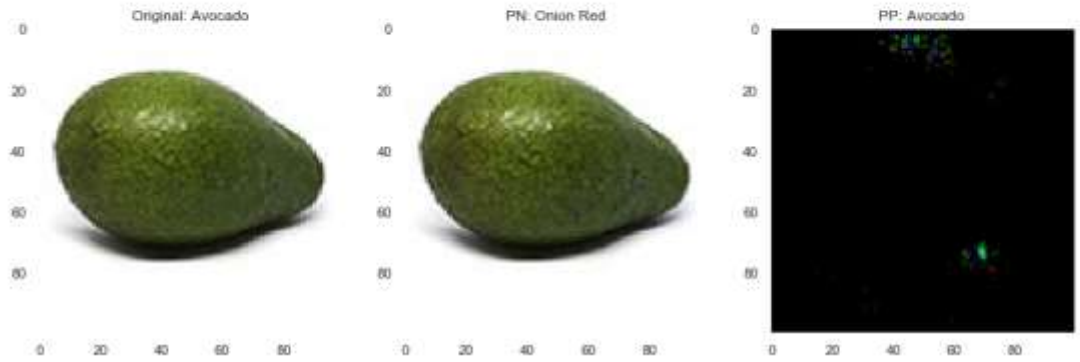
```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

```
cem_pn = CEM(cnn_fruits_mdl, 'PN', (1,) + X_train.shape[1:],\
             feature_range=(0.0, 1.0), max_iterations=100,\
             ae_model=autoencoder_mdl, gamma=100, c_init=1.)
cem_pn.fit(X_train, no_info_type='median')

cem_pp = CEM(cnn_fruits_mdl, 'PP', (1,) + X_train.shape[1:],\
             feature_range=(0.0, 1.0), max_iterations=100,\
             ae_model=autoencoder_mdl, gamma=100, c_init=0.5,\
             beta=0.1)
cem_pp.fit(X_train, no_info_type='median')
```

```
CEM(meta={
    'name': 'CEM',
    'type': ['blackbox', 'tensorflow', 'keras'],
    'explanations': ['local'],
    'params': {
        'mode': 'PP',
        'shape': (1, 100, 100, 3),
        'kappa': 0.0,
        'beta': 0.1,
        'feature_range': (0.0, 1.0),
        'gamma': 100,
        'learning_rate_init': 0.01,
        'max_iterations': 100,
        'c_init': 0.5,
        'c_steps': 10,
        'eps': (0.001, 0.001),
        'clip': (-100.0, 100.0),
        'update_num_grad': 1,
        'no_info_val': None,
        'write_dir': None,
        'is_model': True,
        'is_model_keras': False,
        'is_ae': True,
        'is_ae_keras': False,
        'no_info_type': 'median'
    }
})
```

```
cem_pn_preds = []
cem_pn_imgs = []
cem_pp_preds = []
cem_pp_imgs = []

for i in range(len(labels_l)):
    pn_explanation = cem_pn.explain(np.array([X_tp[i]]))
    if pn_explanation.PN_pred is not None:
        cem_pn_preds.append(fruits_l[pn_explanation.PN_pred])
        cem_pn_imgs.append(pn_explanation.PN[0])
    else:
        cem_pn_preds.append("")
        cem_pn_imgs.append(np.ones(X_train.shape[1:4]))

    pp_explanation = cem_pp.explain(np.array([X_tp[i]]))
    if pp_explanation.PP_pred is not None:
        cem_pp_preds.append(fruits_l[pp_explanation.PP_pred])
        norm_img = (pp_explanation.PP[0] - pp_explanation.PP[0].min()) /\
                (pp_explanation.PP[0].max() - pp_explanation.PP[0].min())
        cem_pp_imgs.append(norm_img)
    else:
        cem_pp_preds.append("")
        cem_pp_imgs.append(np.ones(X_train.shape[1:4]))
```

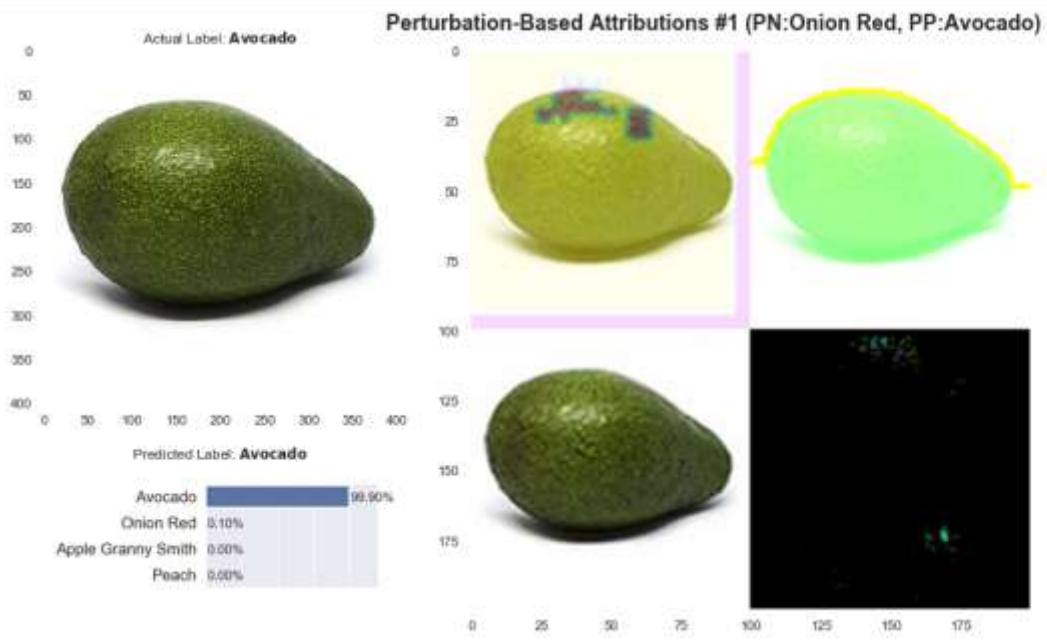
```
plt.subplots(figsize=(15,5))
plt.subplot(1, 3, 1)
plt.imshow(X_tp[0])
plt.grid(b=None)
plt.title('Original: '+fruits_l[labels_l[0]])
plt.subplot(1, 3, 2)
plt.imshow(cem_pn_imgs[0])
plt.grid(b=None)
plt.title('PN: '+cem_pn_preds[0])
plt.subplot(1, 3, 3)
plt.imshow(cem_pp_imgs[0])
plt.grid(b=None)
plt.title('PP: '+cem_pp_preds[0])
plt.show()
```

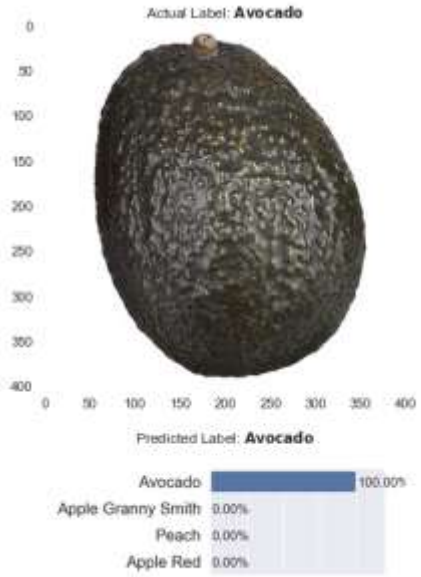


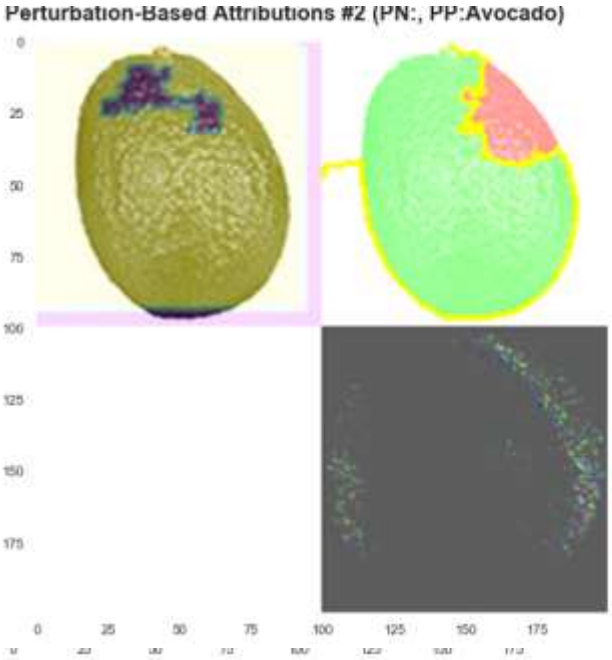
```
np.set_printoptions(precision=3, suppress=True)

pp_pred = cnn_fruits_mdl.predict(np.array([cem_pn_imgs[0]]))[0]
print('PP Probs:\t%s' % pp_pred)
print('PP Pred:\t%s' % fruits_l[pp_pred.argmax()])

pn_pred = cnn_fruits_mdl.predict(np.array([cem_pp_imgs[0]]))[0]
print('PN Probs:\t%s' % pn_pred)
print('PN Pred:\t%s' % fruits_l[pn_pred.argmax()])
```

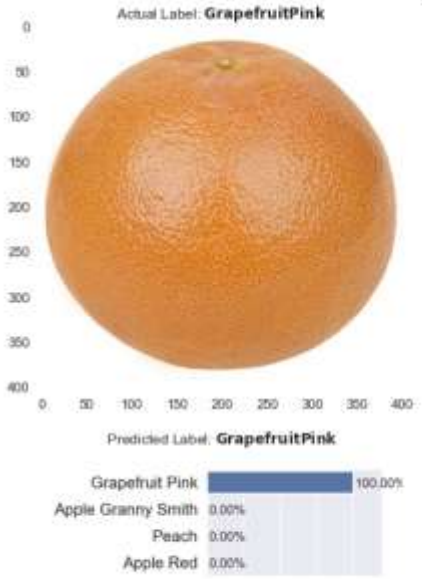
```
PP Probs:       [0.    0.    0.    0.155 0.    0.    0.    0.    0.    0.845 0.    0.
 0.    0.    0.    0.    ]
PP Pred:        Onion Red
PN Probs:       [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
PN Pred:        Avocado
```

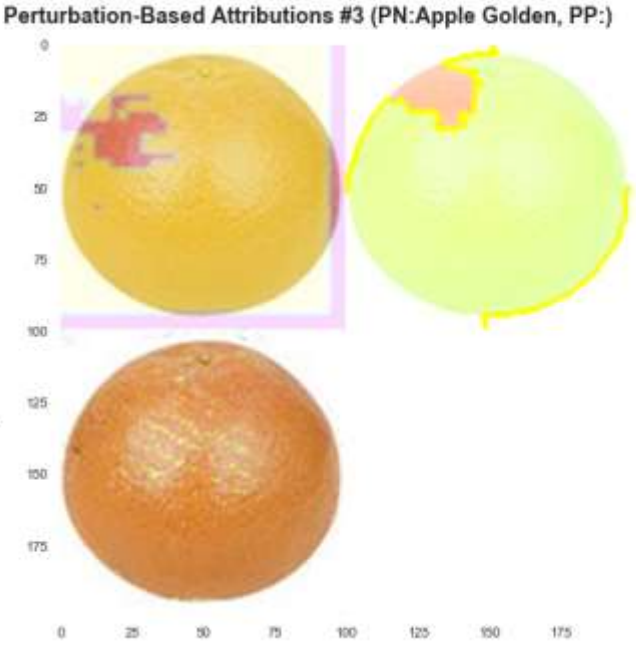
```
for pos, idx in zip([*range(len(idxs))], idxs):
    orig_img = X_val_orig[idx]
    map1 = mldatasets.heatmap_overlay(X_tp[pos], os_maps[pos]/255)
    img_show, mask_show = lime_expl[pos].\
                                get_image_and_mask(lime_expl[pos].top_labels[0],\
                                    positive_only=False, num_features=10)
    map2 = np.uint8(mark_boundaries(img_show / 2 + 0.5, mask_show)[..., :3] * 255)
    map3 = np.uint8(cem_pn_imgs[pos][..., :3] * 255)
    map4 = np.uint8(cem_pp_imgs[pos][..., :3] * 255)
    viz_img = cv2.vconcat([
            cv2.hconcat([map1, map2]),
            cv2.hconcat([map3, map4])
        ])
    y_true = y_val[idx,0]
    y_pred = y_val_pred[idx]
    probs_s = probs_df.loc[idx]
    title = 'Perturbation-Based Attributions #{} (PN:{}, PP:{})'.format(pos+1,\
                                    cem_pn_preds[pos], cem_pp_preds[pos])
    mldatasets.compare_img_pred_viz(orig_img, viz_img, y_true,\
                                    y_pred, probs_s, title=title)
```

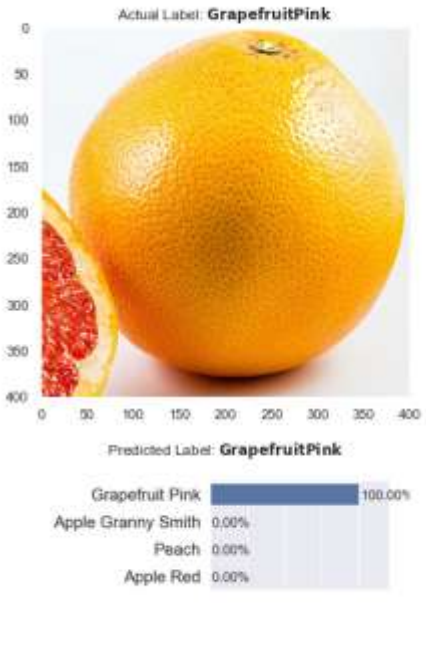


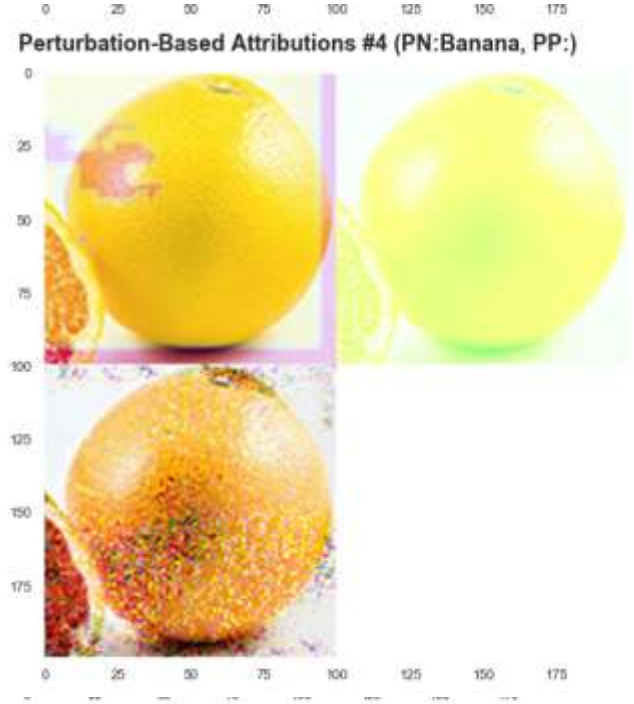

Actual Label: Avocado
Predicted Label: Avocado
Avocado 100.00%
Apple Granny Smith 0.00%
Peach 0.00%
Apple Red 0.00%

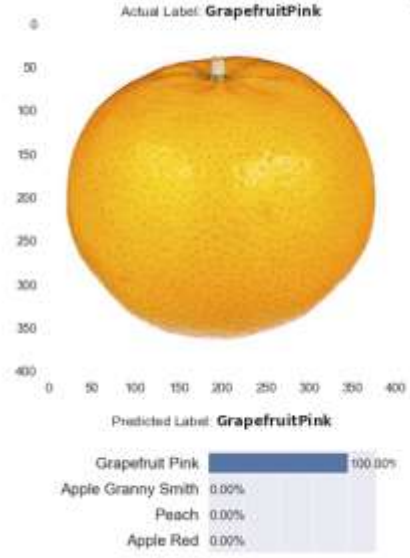

Perturbation-Based Attributions #2 (PN:, PP:Avocado)

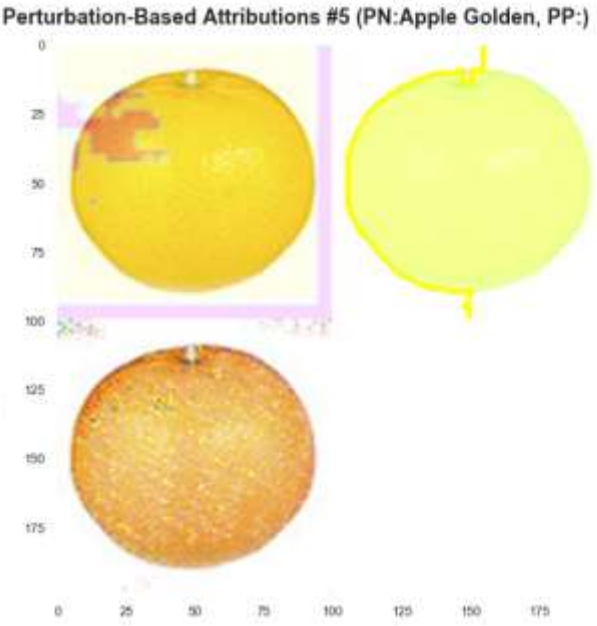

Actual Label: GrapefruitPink
Predicted Label: GrapefruitPink
Grapefruit Pink 100.00%
Apple Granny Smith 0.00%
Peach 0.00%
Apple Red 0.00%

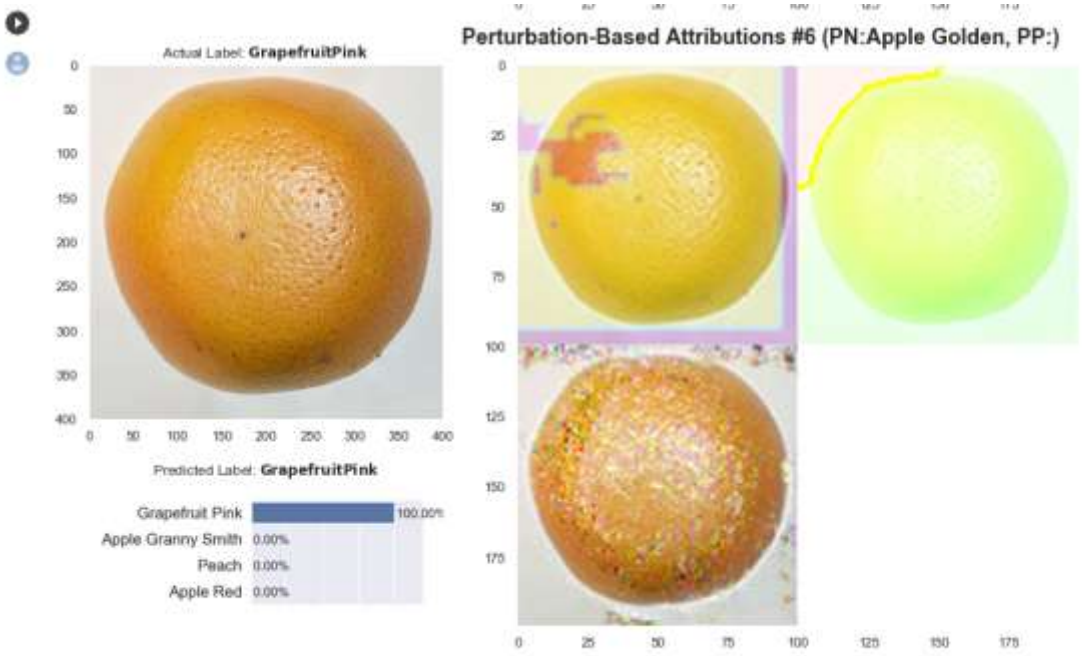

Perturbation-Based Attributions #3 (PN:Apple Golden, PP:)


Actual Label: GrapefruitPink
Predicted Label: GrapefruitPink
Grapefruit Pink 100.00%
Apple Granny Smith 0.00%
Peach 0.00%
Apple Red 0.00%


Perturbation-Based Attributions #4 (PN:Banana, PP:)


Actual Label: GrapefruitPink
Predicted Label: GrapefruitPink
Grapefruit Pink 100.00%
Apple Granny Smith 0.00%
Peach 0.00%
Apple Red 0.00%


Perturbation-Based Attributions #5 (PN:Apple Golden, PP:)




```
background = X_train[np.random.choice(X_train.shape[0], 100, replace=False)]
print(background.shape)
```

```
(100, 100, 100, 3)
```

```
explainer = shap.DeepExplainer(cnn_fruits_mdl, background)
shap_values = explainer.shap_values(X_tp)
```

```
keras is no longer supported, please use tf.keras instead.
```

```
shap.image_plot(shap_values[3] + shap_values[6], X_tp)
```

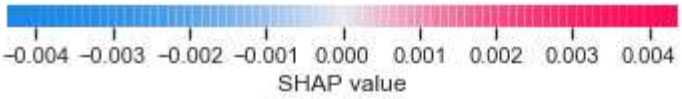




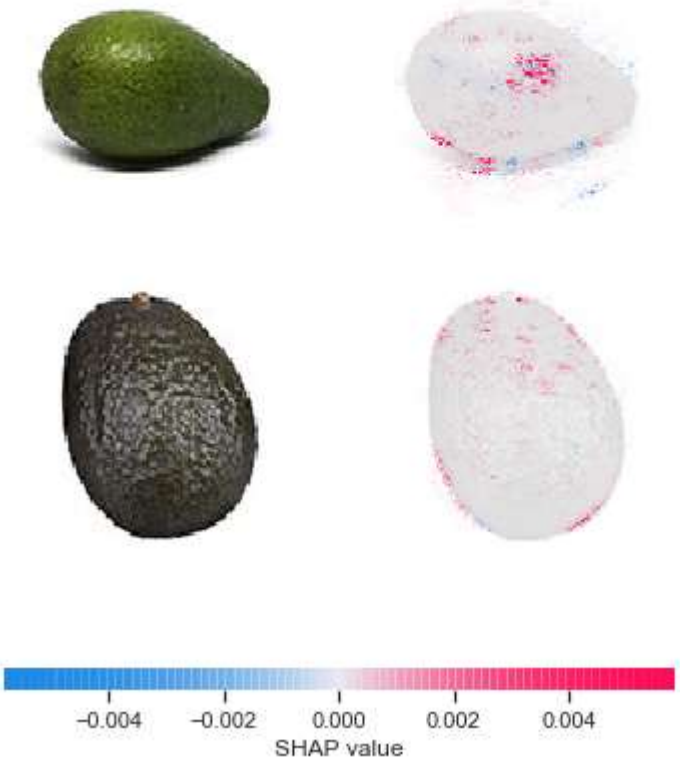









```
]  samp_idx = [0,1]
   samp_shap_values = [s[samp_idx] for s in shap_values]
   shap.image_plot(samp_shap_values[3], X_tp[samp_idx])
```



Chapter 9 (Traffic):

Installing the Libraries

These are all already installed on Google Colab by default so install only if running elsewhere (and ***not already installed***):

```
[ ] !pip install --upgrade pandas numpy scikit-learn statsmodels tensorflow keras matplotlib seaborn
```

Install these if running on Google Colab or ***not already installed***:

```
[ ] !pip install --upgrade machine-learning-datasets
    !pip install --upgrade alibi distython shap SALib
```

Loading the Libraries

```
[ ] import math
    import os
    import machine_learning_datasets as mldatasets
    import pandas as pd
    import numpy as np
    from sklearn.preprocessing import MinMaxScaler
    from sklearn import metrics
    from statsmodels.tsa.seasonal import seasonal_decompose
    from statsmodels.tsa.stattools import acf
    import tensorflow as tf
    from tensorflow import keras
    from tensorflow.keras.preprocessing.sequence import TimeseriesGenerator
    from keras.utils.data_utils import get_file
    import matplotlib.pyplot as plt
    import seaborn as sns
    from alibi.explainers import IntegratedGradients
    from distython import HEOM
    import shap
    from SALib.sample import morris as ms
    from SALib.analyze import morris as ma
    from SALib.plotting import morris as mp
    from SALib.sample import saltelli as ss
```

```
[ ] from SALib.plotting import bar as sp
```

```
    Using plaidml.keras.backend backend.
```

```
[ ] print(tf.__version__)
```

```
    2.2.1
```

Understanding and Preparing the Data

```
[ ] traffic_df = mldatasets.load("traffic-volume", prepare=True)
```

```
    https://archive.ics.uci.edu/ml/machine-learning-databases/00492/Metro_Interstate_Traffic_Volume.csv.gz downloaded to /Users/smasis/Documents/OTHER/InterpretableMLBook/programming/Cha
    1 dataset files found in /Users/smasis/Documents/OTHER/InterpretableMLBook/programming/Chapter9/data folder
    parsing /Users/smasis/Documents/OTHER/InterpretableMLBook/programming/Chapter9/data/Metro_Interstate_Traffic_Volume.csv.gz
```

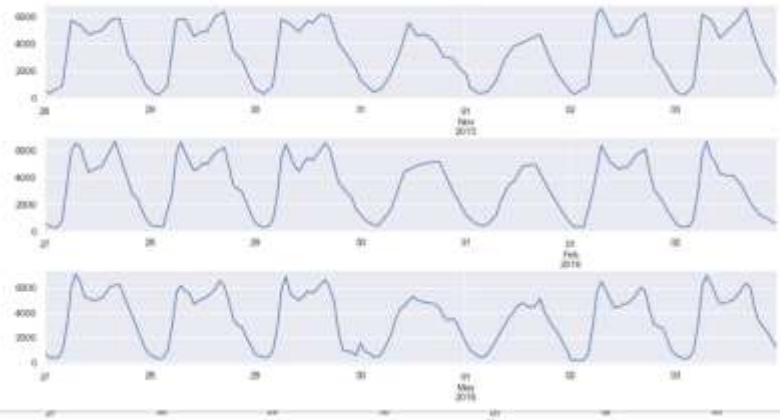
```
[ ] traffic_df.info()
```

```
    <class 'pandas.core.frame.DataFrame'>
    DatetimeIndex: 25656 entries, 2015-10-20 08:00:00 to 2018-09-30 23:00:00
    Data columns (total 15 columns):
     #   Column           Non-Null Count  Dtype
    ---  ------           --------------  -----
     0   dow              25656 non-null  int64
     1   hr               25656 non-null  int64
     2   temp             25656 non-null  float64
     3   rain_1h          25656 non-null  float64
     4   cloud_coverage   25656 non-null  float64
     5   is_holiday       25656 non-null  int64
     6   traffic_volume   25656 non-null  float64
     7   weather_Clear    25656 non-null  uint8
     8   weather_Clouds   25656 non-null  uint8
     9   weather_Haze     25656 non-null  uint8
     10  weather_Mist     25656 non-null  uint8
     11  weather_Other    25656 non-null  uint8
     12  weather_Rain     25656 non-null  uint8
     13  weather_Snow     25656 non-null  uint8
     14  weather_Unknown  25656 non-null  uint8
    dtypes: float64(4), int64(3), uint8(8)
    memory usage: 1.0 MB
```

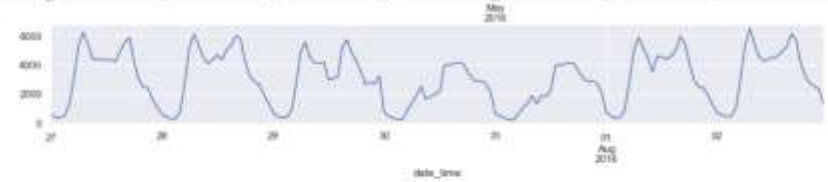
Data Understanding

Understanding Weeks

```
[ ] sns.set()
    fig, (ax0,ax1,ax2,ax3) = plt.subplots(4,1, figsize=(15,8))
    plt.subplots_adjust(top = 0.99, bottom=0.01, hspace=0.4)
    traffic_df[:168].traffic_volume.plot(ax=ax0)
    traffic_df[(168*13):(168*14)].traffic_volume.plot(ax=ax1)
    traffic_df[(168*26):(168*27)].traffic_volume.plot(ax=ax2)
    traffic_df[(168*39):(168*40)].traffic_volume.plot(ax=ax3)
    plt.show()
```

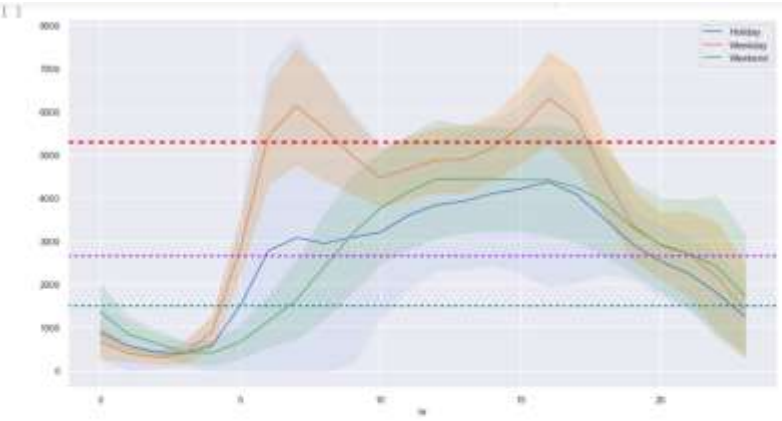


```
[ ]
```



Understanding Days

```
[ ] weekend_df = traffic_df[['hr', 'dow', 'is_holiday', 'traffic_volume']].copy()
    weekend_df['type_of_day'] = np.where(weekend_df.is_holiday == 1, 'Holiday',
                                         np.where(weekend_df.dow >= 5, 'Weekend', 'Weekday'))
    weekend_df = weekend_df.groupby(['type_of_day','hr'])['traffic_volume'].agg(['mean','std']).\
                 reset_index().pivot(index='hr', columns='type_of_day', values=['mean', 'std'])
    weekend_df.columns = ['_'.join(col).strip().replace('mean_','')\
                          for col in weekend_df.columns.values]
    fig, ax = plt.subplots(figsize=(15,8))
    weekend_df[['Holiday','Weekday','Weekend']].plot(ax=ax)
    plt.fill_between(weekend_df.index,\
                     np.maximum(weekend_df.Weekday - 2 * weekend_df.std_Weekday, 0),\
                     weekend_df.Weekday + 2 * weekend_df.std_Weekday,
                     color='darkorange', alpha=0.2)
    plt.fill_between(weekend_df.index,\
                     np.maximum(weekend_df.Weekend - 2 * weekend_df.std_Weekend, 0),\
                     weekend_df.Weekend + 2 * weekend_df.std_Weekend,
                     color='green', alpha=0.1)
    plt.fill_between(weekend_df.index,\
                     np.maximum(weekend_df.Holiday - 2 * weekend_df.std_Holiday, 0),\
                     weekend_df.Holiday + 2 * weekend_df.std_Holiday,
                     color='cornflowerblue', alpha=0.1)
    ax.axhline(y=5300, linewidth=3, color='red', dashes=(2,2))
    ax.axhline(y=2650, linewidth=3, color='darkviolet', dashes=(2,2))
    ax.axhline(y=1500, linewidth=3, color='teal', dashes=(2,2))
    plt.show()
```



Data Preparation

```
train = traffic_df[:-4368]
valid = traffic_df[-4368:-2184]
test = traffic_df[-2184:]
```

```
plt.figure(figsize=(15,4))
plt.plot(train.index.values, train.traffic_volume.values,\
         label='train')
plt.plot(valid.index.values, valid.traffic_volume.values,\
         label='validation')
         label='test')
```

```
plt.ylabel('Traffic Volume')
plt.legend()
plt.show()
```


```
y_scaler = MinMaxScaler()
y_scaler.fit(traffic_df[['traffic_volume']])
X_scaler = MinMaxScaler()
X_scaler.fit(traffic_df.drop(['traffic_volume'], axis=1))
```

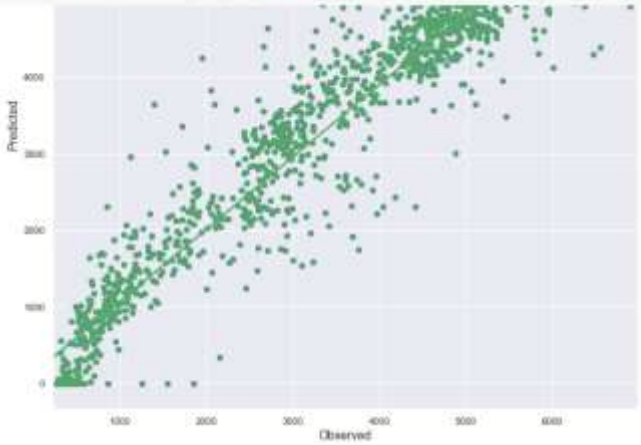
```
MinMaxScaler()
```

```
y_train = y_scaler.transform(train[['traffic_volume']])
X_train = X_scaler.transform(train.drop(['traffic_volume'], axis=1))
y_test = y_scaler.transform(test[['traffic_volume']])
X_test = X_scaler.transform(test.drop(['traffic_volume'], axis=1))
```

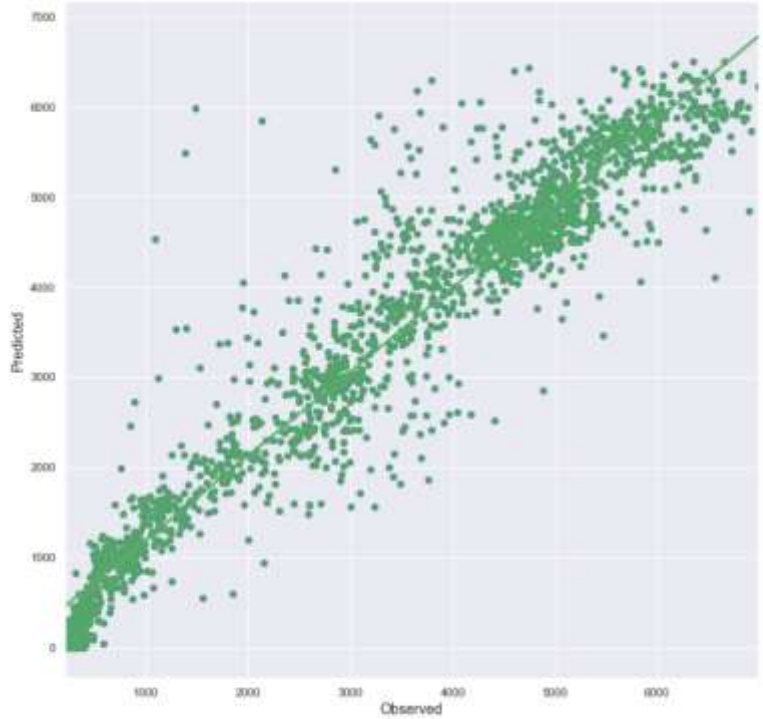
```
gen_train_672 = TimeseriesGenerator(X_train, y_train, length=672, batch_size=24)
gen_test_672 = TimeseriesGenerator(X_test, y_test, length=672, batch_size=24)

print("gen_train_672:\t%s\t*\t%s\t*\t%s" % (len(gen_train_672),\
                                            gen_train_672[0][0].shape,\
                                            gen_train_672[0][1].shape))
print("gen_test_672:\t%s\t*\t%s\t*\t%s" % (len(gen_test_672),\
                                           gen_test_672[0][0].shape,\
```

Using Standard Regression Metrics

```
print(lstm_traffic_672_mdl.name+"\r\n"+("="*100))
y_train_pred_672, y_test_pred_672, y_train_672, y_test_672 =\
    mldatasets.evaluate_reg_mdl(lstm_traffic_672_mdl, gen_train_672, gen_test_672,\
                                y_train, y_test, scaler=y_scaler, y_truncate=True,\
                                predopts={"verbose":1})

print('\r\n'+lstm_traffic_168_mdl.name+"\r\n"+("="*100))
y_train_pred_168, y_test_pred_168, y_train_168, y_test_168 =\
    mldatasets.evaluate_reg_mdl(lstm_traffic_168_mdl, gen_train_168, gen_test_168,\
                                y_train, y_test, scaler=y_scaler, y_truncate=True,\
                                predopts={"verbose":1})
```




```
====================================================================================================
880/880 [==============================] - 17s 19ms/step
84/84 [==============================] - 2s 19ms/step
```



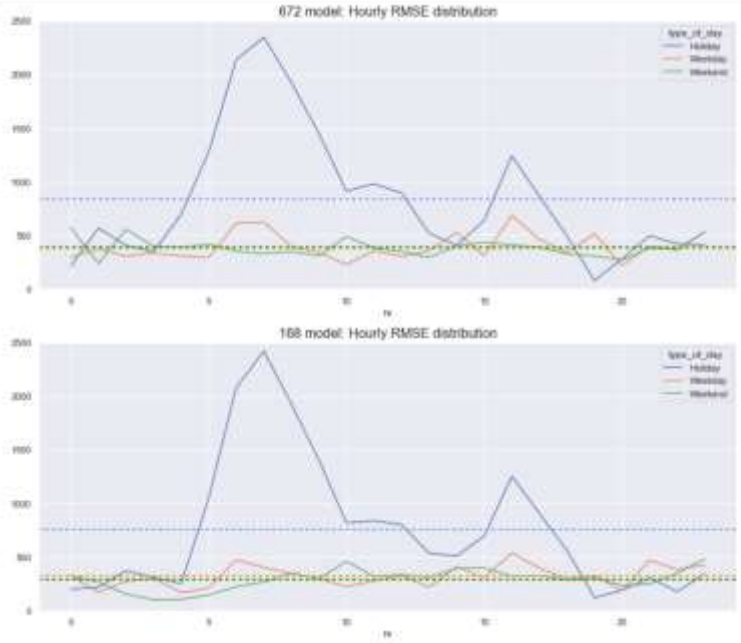

```
RMSE_train: 473.4279    RMSE_test: 561.8984    r2: 0.9187
```

Predictive Error Aggregations

```
evaluate_672_df = test.iloc[-y_test_pred_672.shape[0]:,[0,1,5,6]].\
                  rename(columns={'traffic_volume':'actual_traffic'})
evaluate_672_df['predicted_traffic'] = y_test_pred_672
evaluate_672_df['type_of_day'] = np.where(evaluate_672_df.is_holiday == 1, 'Holiday',
                                 np.where(evaluate_672_df.dow >= 5, 'Weekend', 'Weekday'))
evaluate_672_df.drop(['dow','is_holiday'], axis=1, inplace=True)

evaluate_168_df = test.iloc[-y_test_pred_168.shape[0]:,[0,1,5,6]].\
                  rename(columns={'traffic_volume':'actual_traffic'})
evaluate_168_df['predicted_traffic'] = y_test_pred_168
evaluate_168_df['type_of_day'] = np.where(evaluate_168_df.is_holiday == 1, 'Holiday',
                                 np.where(evaluate_168_df.dow >= 5, 'Weekend', 'Weekday'))
evaluate_168_df.drop(['dow','is_holiday'], axis=1, inplace=True)
```

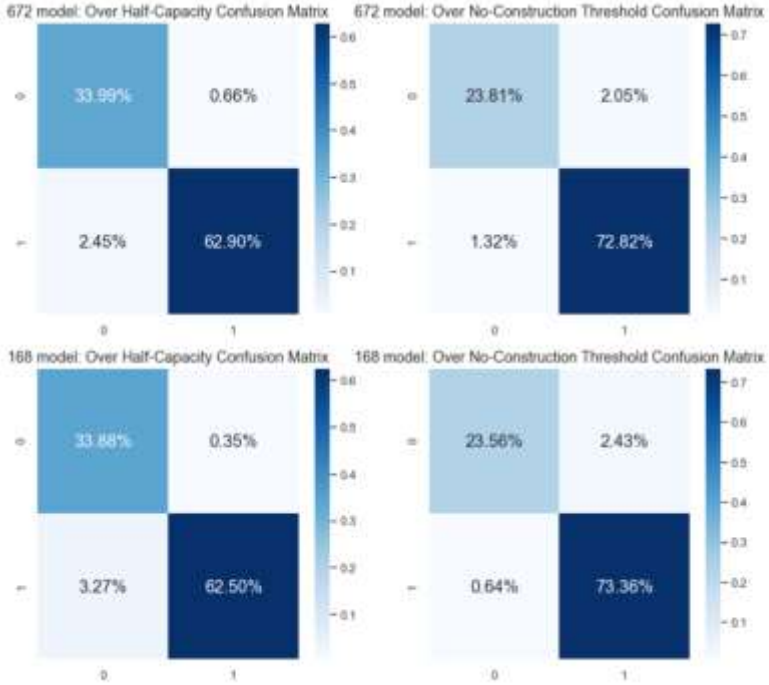
```
evaluate_672_df
```

date_time	hr	actual_traffic	predicted_traffic	type_of_day
2018-07-30 00:00:00	0	640.0	998.761108	Weekday
2018-07-30 01:00:00	1	384.0	64.304565	Weekday
2018-07-30 02:00:00	2	317.0	0.000000	Weekday
2018-07-30 03:00:00	3	333.0	298.437378	Weekday
2018-07-30 04:00:00	4	915.0	1508.385254	Weekday
...	...	...	...	...
2018-09-30 19:00:00	19	3543.0	3202.212646	Weekend
2018-09-30 20:00:00	20	2781.0	2687.730713	Weekend
2018-09-30 21:00:00	21	2159.0	2144.519287	Weekend
2018-09-30 22:00:00	22	1450.0	1600.674072	Weekend
2018-09-30 23:00:00	23	954.0	1190.117798	Weekend



Evaluating It Like A Classification Problem

```
[ ] evaluate_dfs = [evaluate_672_df, evaluate_168_df]
    lookbacks = [672, 168]

    for e in range(2):
        evaluate_df = evaluate_dfs[e]
        lb = lookbacks[e]

        actual_over_half_cap = np.where(evaluate_df['actual_traffic'] >\
                                        2650, 1, 0)
        pred_over_half_cap = np.where(evaluate_df['predicted_traffic'] >\
                                      2650, 1, 0)
        actual_over_nc_thresh = np.where(evaluate_df['actual_traffic'] >\
                                         1500, 1, 0)
        pred_over_nc_thresh = np.where(evaluate_df['predicted_traffic'] >\
                                       1500, 1, 0)

        eldatasets.\
            cmpare_confusion_matrices(actual_over_half_cap, pred_over_half_cap,\
                                      actual_over_nc_thresh, pred_over_nc_thresh,\
                                      str(lb)+' model: Over Half-Capacity',\
                                      str(lb)+' model: Over No-Construction Threshold')
```



Generating LSTM Attributions with Integrated Gradients

```
[ ] y_all = y_scaler.transform(traffic_df[['traffic_volume']])
    X_all = X_scaler.transform(traffic_df.drop(['traffic_volume'], axis=1))
    gen_all_672 = TimeseriesGenerator(X_all, y_all, length=672, batch_size=24)
    gen_all_168 = TimeseriesGenerator(X_all, y_all, length=168, batch_size=24)
    print("gen_all_678:\t%s\t*\t%s\t->\t%s" % (len(gen_all_672),\
                                    gen_all_672[0][0].shape,\
                                    gen_all_672[0][1].shape))
    print("gen_all_168:\t%s\t*\t%s\t->\t%s" % (len(gen_all_168),\
                                    gen_all_168[0][0].shape,\
                                    gen_all_168[0][1].shape))
```

```
gen_all_678:    1041    *       (24, 672, 14)   ->      (24, 1)
gen_all_168:    1062    *       (24, 168, 14)   ->      (24, 1)
```

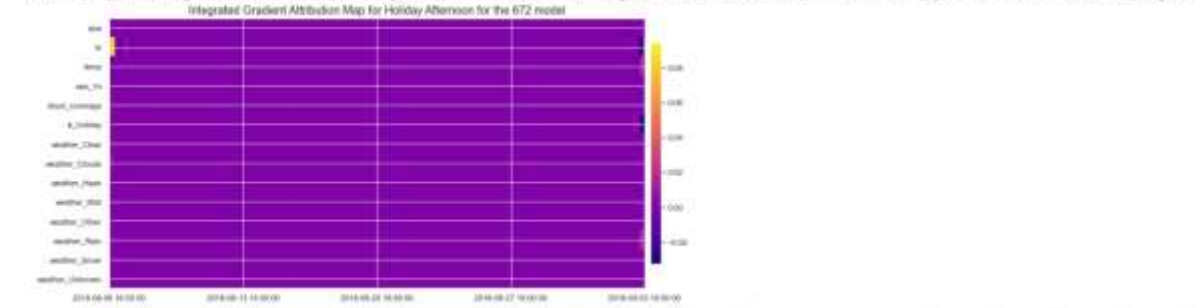
```
[ ] X_df = traffic_df.drop(['traffic_volume'], axis=1).reset_index(drop=True)

    holiday_afternoon_s = X_df[(X_df.index >= 23471) & (X_df.dow==0) &\
                               (X_df.hr==16) & (X_df.is_holiday==1)]
    peak_morning_s = X_df[(X_df.index >= 23471) & (X_df.dow==2) &\
                          (X_df.hr==8) & (X_df.weather_Clouds==1) &\
                          (X_df.temp<20)]
    hot_saturday_s = X_df[(X_df.index >= 23471) & (X_df.dow==5) &\
                          (X_df.hr==12) & (X_df.temp>29)]
```

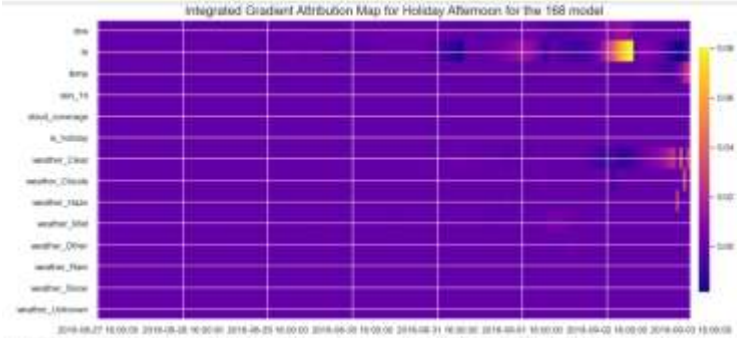
```
[ ] ig_672  = IntegratedGradients(lstm_traffic_672_mdl,\
                          n_steps=25, internal_batch_size=24)
    ig_168  = IntegratedGradients(lstm_traffic_168_mdl,\
                          n_steps=25, internal_batch_size=24)
```

```
[ ] nidx = holiday_afternoon_s.index.tolist()[0] - 672
    batch_X = gen_all_672[nidx//24][0]
    print(batch_X.shape)
```

```
(24, 672, 14)
```

```
[ ]
    end_date = traffic_df.iloc[samples[x].index[-\
                        1:].to_pydatetime()[0]
    date_range = pd.date_range(end=end_date, periods=q,\
                    freq='1').to_pydatetime().tolist()
    columns = samples[x].columns.tolist()

    plt.figure(figsize=(14,7))
    plt.title('Integrated Gradient Attribution Map for {} for the {} model'.\
              format(sample_names[x], lb), fontsize=16)
    plt.imshow(attribution_img, interpolation='nearest',\
                                aspect='auto', cmap='plasma')
    plt.xticks(np.linspace(0,lb,p).astype(int), labels=date_range)
    plt.yticks([*range(14)], labels=columns)
    plt.colorbar(pad=0.05,fraction=0.02,anchor=(1.0,0.0))
    plt.show()
```

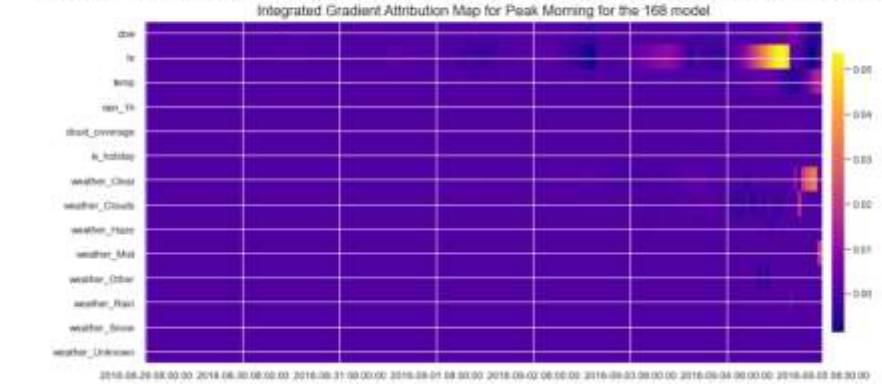





```
It looks like you are passing a model with a scalar output and target is set to `None`.If your model is a regressive model this will produce correct attributions. If your model is a classification m
```


```
[ ]
```

```
It looks like you are passing a model with a scalar output and target is set to `None`.If your model is a regression model this will produce correct attributions. If yo
```




```
CPU times: user 7min 7s, sys: 57.8 s, total: 8min 5s
Wall time: 2min 32s
```

Computing Global and Local Attributions with SHAP's KernelExplainer

Define Filter Function

```
[ ] def filt_fn(X_df, x, lookback):
        #print(x.tolist())
        x_ = x.copy()
        x_[0] = round(x_[0])
        x_[1] = round(x_[1])
        x_[4] = round(x_[4])
        x_[5] = round(x_[5])
        if x_[1] < 0:
            x_[1] = 24 + x_[1]
            x_[0] = x_[0] - 1
        if x_[0] < 0:
            x_[0] = 7 + x_[0]
        x_[6] = round(x_[6])
        x_[7] = round(x_[7])
        x_[8] = round(x_[8])
        x_[9] = round(x_[9])
        x_[10] = round(x_[10])
        x_[11] = round(x_[11])
        x_[12] = round(x_[12])
        x_[13] = round(x_[13])
        #if not (x==x_).all(): print('\t%s' % x_.tolist())
        X_filt_df = X_df[(X_df.index >= lookback) & (X_df.dow==x_[0]) & (X_df.hr==x_[1]) &\
                (X_df.is_holiday==x_[5]) & (X_df.temp-5<=x_[2]) & (X_df.temp+5>=x_[2])]
        if X_filt_df.shape[0]==0:
            X_filt_df = X_df[(X_df.index >= lookback) & (X_df.dow==x_[0]) & (X_df.hr==x_[1]) &\
                            (X_df.is_holiday==x_[5]) & (X_df.temp-8<=x_[2]) & (X_df.temp+8>=x_[2])]
        if X_filt_df.shape[0]==0:
            X_filt_df = X_df[(X_df.index >= lookback) & (X_df.dow==x_[0]) & (X_df.hr==x_[1]) &\
                            (X_df.temp-11<=x_[2]) & (X_df.temp+11>=x_[2])]
        return X_filt_df, x_
```

Define Distance Function

```
[ ] cat_idxs = np.where(traffic_df.drop(['traffic_volume'], axis=1).dtypes != np.float64)[0]
    heom_dist = HEOM(X_df.values, cat_idxs)
```

```
[ ] print(cat_idxs)
```

```
[ 0  1  5  6  7  8  9 10 11 12 13]
```

Define Predict Function

```
[ ] predict_fn = lambda X: mldatasets.\
                        approx_predict_ts(X, X_df, gen_all, lstm_traffic_mdl,\
                              dist_metric=heom_dist.heom, lookback=lookback,\
                              filt_fn=filt_fn, X_scaler=X_scaler, y_scaler=y_scaler)
```

Define Scope for Attributions

```
[ ] working_season_df = traffic_df[lookback:].drop(['traffic_volume'], axis=1).copy()
    working_season_df = working_season_df[(working_season_df.index.month >= 5) &\
                                          (working_season_df.index.month <= 10)]
    explainer = shap.KernelExplainer(predict_fn,\
                                     shap.kmeans(working_season_df.values, 10))
```

Computing the SHAP Values

```
[ ] X_samp_df = working_season_df.sample(48, random_state=rand)
    shap_values = explainer.shap_values(X_samp_df, nsamples=5)
    shap.summary_plot(shap_values, X_samp_df)
```

Show hidden output

Visualize Summary Plots

```
datapoints = [holiday_afternoon_x, peak_morning_x, hot_saturday_x]
datapoint_labels = ['Holiday Afternoon', 'Peak Morning', 'Hot Saturday']
for i in range(len(datapoints)):
    print(datapoint_labels[i])
    shap_values_single = explainer.shap_values(datapoints[i], nsamples=60)
    shap.force_plot(explainer.expected_value, shap_values_single[0],\
                    datapoints[i], matplotlib=True)
    plt.show()
```

```
Holiday Afternoon
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1.0), HTML(value='')))
```


```
Peak Morning
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1.0), HTML(value='')))
```


```
Hot Saturday
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1.0), HTML(value='')))
```

Identifying Influential Features with Factor Prioritization

Computing Morris Sensitivity Indices

```
working_hrs_df = working_season_df[(working_season_df.dow < 5) &\
                                   ((working_season_df.hr < 5) |\
                                    (working_season_df.hr > 12))]
working_hrs_df.describe(percentiles=[.025,.5,.975]).transpose()
```

	count	mean	std	min	2.5%	50%	97.5%	max
dow	2232.0	1.991935	1.415458	0.00	0.00000	2.000	4.00000	4.000
hr	2232.0	5.500000	7.933780	0.00	0.00000	2.500	23.00000	23.000
temp	2232.0	16.026438	5.380406	-2.57	3.17875	16.935	24.47675	30.458
rain_1h	2232.0	0.099628	0.603634	0.00	0.00000	0.000	1.45125	10.920
cloud_coverage	2232.0	29.178763	36.701417	0.00	0.00000	1.000	90.00000	100.000
is_holiday	2232.0	0.037634	0.190353	0.00	0.00000	0.000	1.00000	1.000
weather_Clear	2232.0	0.432348	0.495613	0.00	0.00000	0.000	1.00000	1.000
weather_Clouds	2232.0	0.207885	0.405885	0.00	0.00000	0.000	1.00000	1.000
weather_Haze	2232.0	0.010753	0.103159	0.00	0.00000	0.000	0.00000	1.000
weather_Mist	2232.0	0.104391	0.305835	0.00	0.00000	0.000	1.00000	1.000
weather_Other	2232.0	0.058244	0.234256	0.00	0.00000	0.000	1.00000	1.000
weather_Rain	2232.0	0.181452	0.385478	0.00	0.00000	0.000	1.00000	1.000
weather_Snow	2232.0	0.002240	0.047288	0.00	0.00000	0.000	0.00000	1.000
weather_Unknown	2232.0	0.002688	0.051789	0.00	0.00000	0.000	0.00000	1.000

```
[ ] morris_problem = {
        # There are nine variables
        'num_vars': 9,
        # These are their names
        'names': ['dow', 'hr', 'temp', 'rain_1h',\
                  'cloud_coverage', 'is_holiday',\
                  'weather_Clear', 'weather_Clouds',\
                  'weather_Rain'],
        # Plausible ranges over which we'll move the variables
        'bounds': [[0, 6], # dow
                   [-1, 4], # hr
                   [3., 25.], # temp (C)
                   [0., 1.5], # rain_1h
                   [0., 90.], # cloud_coverage
                   [0, 1], # is_holiday
                   [0, 1], # weather_Clear
                   [0, 1], # weather_Clouds
                   [0, 1] # weather_Rain
                  ],
        # Only weather is grouped together
        'groups': ['dow', 'hr', 'temp', 'rain_1h',\
                   'cloud_coverage', 'is_holiday',\
                    'weather', 'weather', 'weather']
    }
```

```
[ ] morris_sample = ms.sample(morris_problem, 300,\
                              num_levels=4, seed=rand)
    print(morris_sample.shape)
```

```
(2400, 9)
```

```
[ ] morris_sample_mod = np.hstack((morris_sample[:,0:8],\
                        np.zeros((morris_sample.shape[0],1)),\
                        morris_sample[:,8:9],\
                        np.zeros((morris_sample.shape[0],2))))
    print(morris_sample_mod.shape)
```

```
(2400, 14)
```

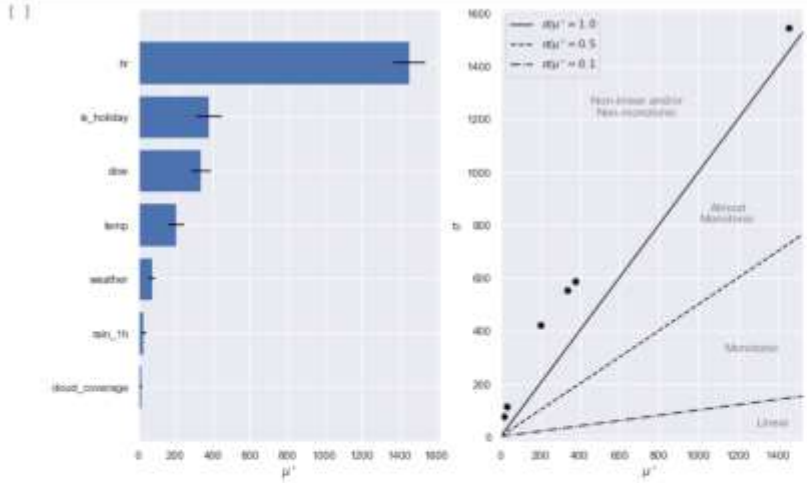
Analyzing the Elementary Effects

```
[ ] morris_df = pd.DataFrame({'features':morris_sensitivities['names'],\
                              'μ':morris_sensitivities['mu'],\
                              'μ*':morris_sensitivities['mu_star'],\
                              'σ':morris_sensitivities['sigma']})
    morris_df.sort_values('μ*', ascending=False).style.\
        background_gradient(cmap='plasma', subset=['μ*'])
```

	features	μ	μ*	σ
1	hr	-429.299988	1455.509888	1544.544312
5	is_holiday	-345.794891	379.520447	588.769897
0	dow	130.311523	338.988451	554.438880
2	temp	62.087780	202.984329	422.309845
6	weather	nan	75.732839	nan
3	rain_1h	-2.807379	30.730110	113.282100
4	cloud_coverage	9.897465	17.152824	74.320000

```
[ ] fig, (ax0, ax1) = plt.subplots(1,2, figsize=(12,8))
    mp.horizontal_bar_plot(ax0, morris_sensitivities, {})
    mp.covariance_plot(ax1, morris_sensitivities, {})
    ax1.text(ax1.get_xlim()[1]*0.45, ax1.get_ylim()[1]*0.75,\
             'Non-linear and/or\nNon-monotonic',\
             horizontalalignment='center', color='gray')
    ax1.text(ax1.get_xlim()[1]*0.75, ax1.get_ylim()[1]*0.5,\
             'Almost\nMonotonic', horizontalalignment='center',\
             color='gray')
    ax1.text(ax1.get_xlim()[1]*0.83, ax1.get_ylim()[1]*0.2,\
             'Monotonic', horizontalalignment='center', color='gray')
    ax1.text(ax1.get_xlim()[1]*0.9, ax1.get_ylim()[1]*0.025,\
             'linear', horizontalalignment='center', color='gray')
    plt.show()
```

[ ]



Quantifying Uncertainty and Cost-Sensitivity with Factor Fixing

Generating and Predicting on Salteli Samples

```
[ ] sobol_problem = {
        'num_vars': 7,
        'names': ['dow', 'hr', 'temp', 'rain_1h',\
                  'cloud_coverage', 'is_holiday',\
                  'weather_Clear'],
        'bounds': [[0, 4], # dow
```

Performing a Sobol Sensitivity Analysis

```
[ ] print(max(saltelli_preds[:,0]))
```

```
1879.7686
```

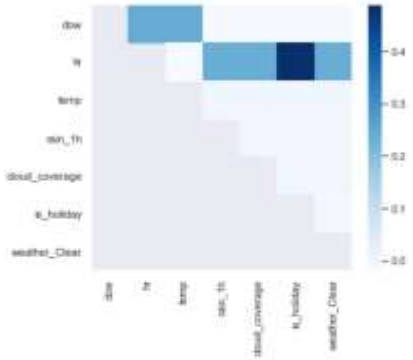
```
[ ] costs = np.where(saltelli_preds > 1500, 1,0)[:,0]
    factor_fixing_sa = sa.analyze(sobol_problem, costs,\
                          calc_second_order=True, print_to_console=False)
```

```
[ ] sobol_df = pd.DataFrame({'features':sobol_problem['names'],\
                             '1st':factor_fixing_sa['S1'],\
                             'Total':factor_fixing_sa['ST'],\
                             'Total Conf':factor_fixing_sa['ST_conf'],\
                             'Mean of Input':saltelli_sample.mean(axis=0)[0:7]})
    sobol_df.sort_values('Total', ascending=False).style.\
        background_gradient(cmap='plasma', subset=['Total'])
```

	features	1st	Total	Total Conf	Mean of Input
1	hr	0.009185	0.886824	0.912979	1.496831
2	temp	0.006123	0.606757	0.660847	14.059766
0	dow	0.009185	0.380068	0.388337	1.996599
5	is_holiday	0.003062	0.380068	0.479628	0.496047
6	weather_Clear	-0.003062	0.126689	0.314201	0.496023
3	rain_1h	0.000000	0.000000	0.000000	5.511458
4	cloud_coverage	0.000000	0.000000	0.000000	50.024740

```
[ ] plt.figure(figsize=(6, 5))
    sns.heatmap(factor_fixing_sa['S2'], cmap='Blues',\
                xticklabels=sobol_problem['names'],\
                yticklabels=sobol_problem['names'])
    plt.show()
```

```
[ ] plt.figure(figsize=(6, 5))
    sns.heatmap(factor_fixing_sa['S2'], cmap='Blues',\
                xticklabels=sobol_problem['names'],\
                yticklabels=sobol_problem['names'])
    plt.show()
```



Incorporating a Realistic Cost Function

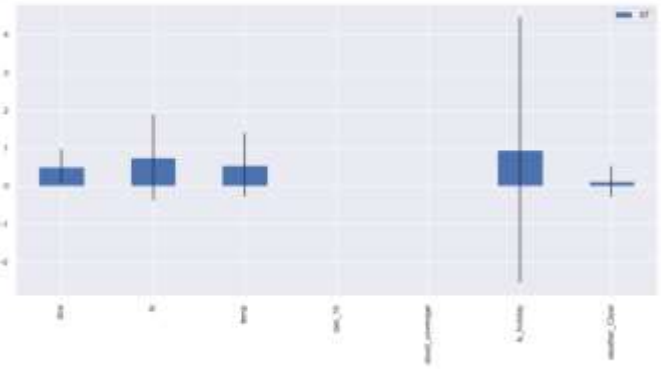
```
[ ] #Join input and outputs into a sample+prediction array
    saltelli_sample_preds = np.hstack((saltelli_sample, saltelli_preds))
```

```
[ ] #Define cost function
    def cost_fn(x):
        cost = 0
        if x[7] > 1500:
            cost = (x[7] - 1500) * 15
            if round(x[1]) == 4:
```

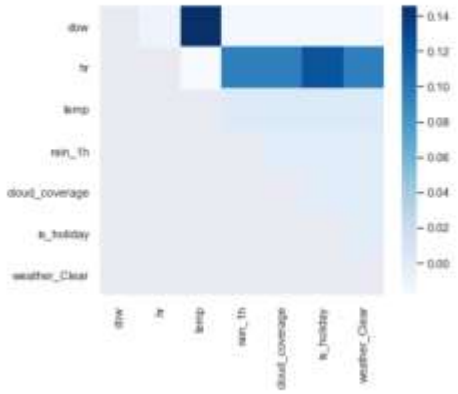
```
[ ]
```

	features	1st	Total	Total Conf	Mean of Input
5	is_holiday	0.000652	0.953693	3.026314	0.498047
1	hr	0.010101	0.748596	1.132660	1.495931
2	temp	0.000677	0.553399	0.843213	14.059769
0	dow	0.000674	0.519429	0.452778	1.995099
6	weather_Clear	-0.002778	0.521222	0.404680	0.499023
3	rain_1h	0.000000	0.000000	0.000000	5.511458
4	cloud_coverage	0.000000	0.000000	0.000000	50.024740

```
[ ] factor_fixing2_df = factor_fixing2_sa.to_df()
    fig, (ax) = plt.subplots(1,1, figsize=(15, 7))
    sn.plot(factor_fixing2_df[0], ax=ax)
    plt.show()
```



```
[ ] plt.figure(figsize=(6, 5))
    sns.heatmap(factor_fixing2_sa['S2'], cmap='Blues',\
                xticklabels=sobol_problem['names'],\
                yticklabels=sobol_problem['names'])
    plt.show()
```



Chapter 10 (Mailer):

```
[1] !pip install --upgrade pandas numpy tqdm scikit-learn mlxtend scipy xgboost matplotlib seaborn
      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.4/9.4 MB 30.9 MB/s eta 0:00:00
    Requirement already satisfied: seaborn in /usr/local/lib/python3.8/dist-packages (0.11.2)
    Collecting seaborn
      Downloading seaborn-0.12.2-py3-none-any.whl (293 kB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 293.3/293.3 kB 18.4 MB/s eta 0:00:00
    Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2022.7.1)
    Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2.8.2)
    Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (3.1.0)
    Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.2.0)
    Requirement already satisfied: setuptools in /usr/local/lib/python3.8/dist-packages (from mlxtend) (57.4.0)
    Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (23.0)
    Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (1.4.4)
    Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (7.1.2)
    Requirement already satisfied: pyparsing>=2.2.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (3.0.9)
    Collecting fonttools>=4.22.0
      Downloading fonttools-4.38.0-py3-none-any.whl (965 kB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 965.4/965.4 kB 25.7 MB/s eta 0:00:00
    Collecting contourpy>=1.0.1
      Downloading contourpy-1.0.7-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (300 kB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 300.0/300.0 kB 4.8 MB/s eta 0:00:00
    Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (0.11.0)
    Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.8.1->pandas) (1.15.0)
    Installing collected packages: numpy, fonttools, scipy, pandas, contourpy, xgboost, scikit-learn, matplotlib, seaborn, mlxtend
      Attempting uninstall: numpy
        Found existing installation: numpy 1.21.6
        Uninstalling numpy-1.21.6:
          Successfully uninstalled numpy-1.21.6
      Attempting uninstall: scipy
        Found existing installation: scipy 1.7.3
        Uninstalling scipy-1.7.3:
          Successfully uninstalled scipy-1.7.3
      Attempting uninstall: pandas
        Found existing installation: pandas 1.3.5
        Uninstalling pandas-1.3.5:
          Successfully uninstalled pandas-1.3.5
      Attempting uninstall: xgboost
        Found existing installation: xgboost 0.90
        Uninstalling xgboost-0.90:
          Successfully uninstalled xgboost-0.90
      Attempting uninstall: scikit-learn
        Found existing installation: scikit-learn 1.0.2
        Uninstalling scikit-learn-1.0.2:
          Successfully uninstalled scikit-learn-1.0.2
```

```
!pip install --upgrade machine-learning-datasets
!pip install --upgrade shap sklearn-genetic
```


```
  Downloading attrs-20.3.0-py2.py3-none-any.whl (49 kB)
     |████████████████████████████████| 49 kB 1.2 MB/s eta 0:00:00
Requirement already satisfied: dill>=0.3.0 in /usr/local/lib/python3.6/dist-packages (from slds>=0.0.0->machine-learning-datasets) (0.3.0)
Collecting tensorflow~=2.5.0
  Downloading tensorflow-2.5.0-cp38-cp38-manylinux2010_x86_64.whl (394.6 MB)
     |████████████████████████████████| 394.6 MB 3.8 MB/s eta 0:00:00
Requirement already satisfied: Pillow>=6.0.0 in /usr/local/lib/python3.6/dist-packages (from slds>=0.0.0->machine-learning-datasets) (7.1.2)
Requirement already satisfied: spacy[lookups]>=3.0.0 in /usr/local/lib/python3.6/dist-packages (from slds>=0.0.0->machine-learning-datasets) (3.0.4)
Requirement already satisfied: scikit-image>=0.17.1 in /usr/local/lib/python3.6/dist-packages (from slds>=0.0.0->machine-learning-datasets) (0.18.1)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=3.2.2->machine-learning-datasets) (20.8)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=3.2.2->machine-learning-datasets) (4.38.0)
Requirement already satisfied: cmdstanpy>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=3.2.2->machine-learning-datasets) (1.0.7)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=3.2.2->machine-learning-datasets) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=3.2.2->machine-learning-datasets) (1.4.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=3.2.2->machine-learning-datasets) (0.11.0)
Requirement already satisfied: pyparsing>=2.2.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=3.2.2->machine-learning-datasets) (3.0.9)
Requirement already satisfied: setuptools>=1 in /usr/local/lib/python3.6/dist-packages (from mlxtend>=0.14.0->machine-learning-datasets) (57.4.0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.6/dist-packages (from pandas>=1.0.0->machine-learning-datasets) (2022.7.1)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from patsy>=0.5.1->machine-learning-datasets) (1.15.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn>=0.23.0->machine-learning-datasets) (1.2.0)
Requirement already satisfied: patsy>=0.5.0 in /usr/local/lib/python3.6/dist-packages (from statsmodels>=0.11.0->machine-learning-datasets) (0.5.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests>=2.0.0->slds>=0.0.0->machine-learning-datasets) (2.10)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.6/dist-packages (from requests>=2.0.0->slds>=0.0.0->machine-learning-datasets) (2.0.12)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests>=2.0.0->slds>=0.0.0->machine-learning-datasets) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests>=2.0.0->slds>=0.0.0->machine-learning-datasets) (2022.12.7)
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image>=0.17.1->slds>=0.0.0->machine-learning-datasets) (2.9.0)
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.6/dist-packages (from scikit-image>=0.17.1->slds>=0.0.0->machine-learning-datasets) (2021.1.11.1)
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.6/dist-packages (from scikit-image>=0.17.1->slds>=0.0.0->machine-learning-datasets) (1.4.1)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image>=0.17.1->slds>=0.0.0->machine-learning-datasets) (2.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.6/dist-packages (from spacy[lookups]>=3.0.0->slds>=0.0.0->machine-learning-datasets) (2.11.3)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.6/dist-packages (from spacy[lookups]>=3.0.0->slds>=0.0.0->machine-learning-datasets) (1.0.9)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from spacy[lookups]>=3.0.0->slds>=0.0.0->machine-learning-datasets) (3.0.8)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.10 in /usr/local/lib/python3.6/dist-packages (from spacy[lookups]>=3.0.0->slds>=0.0.0->machine-learning-datasets) (3.0.12)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.6/dist-packages (from spacy[lookups]>=3.0.0->slds>=0.0.0->machine-learning-datasets) (3.3.0)
Requirement already satisfied: pathy>=0.3.5 in /usr/local/lib/python3.6/dist-packages (from spacy[lookups]>=3.0.0->slds>=0.0.0->machine-learning-datasets) (0.10.1)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.6/dist-packages (from spacy[lookups]>=3.0.0->slds>=0.0.0->machine-learning-datasets) (2.0.7)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.6/dist-packages (from spacy[lookups]>=3.0.0->slds>=0.0.0->machine-learning-datasets) (1.0.4)
Requirement already satisfied: wasabi<1.1.0,>=0.8.1 in /usr/local/lib/python3.6/dist-packages (from spacy[lookups]>=3.0.0->slds>=0.0.0->machine-learning-datasets) (0.10.1)
Requirement already satisfied: srsly<3.0.0,>=2.4.1 in /usr/local/lib/python3.6/dist-packages (from spacy[lookups]>=3.0.0->slds>=0.0.0->machine-learning-datasets) (2.4.5)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.11.0,>=1.7.4 in /usr/local/lib/python3.6/dist-packages (from spacy[lookups]>=3.0.0->slds>=0.0.0->machine-learning-datasets) (1.10.4)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.6/dist-packages (from spacy[lookups]>=3.0.0->slds>=0.0.0->machine-learning-datasets) (2.0.8)
Requirement already satisfied: thinc<8.1.0,>=8.0.3 in /usr/local/lib/python3.6/dist-packages (from spacy[lookups]>=3.0.0->slds>=0.0.0->machine-learning-datasets) (8.1.7)
```


```
[ ] import math
    import os
    import machine_learning_datasets as mldatasets
    import pandas as pd
    import numpy as np
    # Understanding The Effect of Irrelevant Features
    import timeit
    from tqdm.notebook import tqdm
    # Filter-Based Methods
    from sklearn.feature_selection import VarianceThreshold, mutual_info_classif, SelectKBest
    # Embedded Methods
    from sklearn.feature_selection import SelectFromModel
    from sklearn.linear_model import LogisticRegression, LassoCV, LassoLarsCV, LassoLarsIC
    # Wrapper Methods
    from mlxtend.feature_selection import SequentialFeatureSelector
    # Hybrid Methods
    from sklearn.feature_selection import RFECV
    # Advanced Methods
    from sklearn.decomposition import PCA
    import shap
    from genetic_selection import GeneticSelectionCV
    from scipy.stats import rankdata
    # Models
    from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
    from sklearn.ensemble import ExtraTreesRegressor, RandomForestRegressor
    import xgboost as xgb
    # Visualize
    import matplotlib.pyplot as plt
    import seaborn as sns
```

Understanding and Preparing the Data

```
[ ] X_train, X_test, y_train, y_test = mldatasets.load("nonprofit-mailer", prepare=True)
    y_train = y_train.squeeze()
    y_test = y_test.squeeze()
```


```
https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python/raw/master/datasets/mailing-list.zip downloaded to /Users/smasis/Documents/OTHE
/Users/smasis/Documents/OTHER/InterpretableMLBook/programming/Chapter10/data/mailing-list.zip uncompressed to /Users/smasis/Documents/OTHER/InterpretableMLBo
3 dataset files found in /Users/smasis/Documents/OTHER/InterpretableMLBook/programming/Chapter10/data/mailing-list folder
```

```
[ ] print(X_train.shape)
    print(y_train.shape)
    print(X_test.shape)
    print(y_test.shape)
```

```
(95485, 435)
(95485,)
(96017, 435)
(96017,)
```

```
[ ] var_cost = 0.68

    y_test_donators = y_test[y_test > 0]
    test_donators = len(y_test_donators)
    test_donations = sum(y_test_donators)
    test_min_profit = test_donations - (len(y_test)*var_cost)
    test_max_profit = test_donations - (test_donators*var_cost)
    print('%s test donators totalling $%.0f (min profit: $%.0f, max proFit: $%.0f)' %\
          (test_donators, test_donations, test_min_profit, test_max_profit))
```

```
4894 test donators totalling $76464 (min profit: $11173, max profit: $73136)
```

```
[ ] y_train_donators = y_train[y_train > 0]
    train_donators = len(y_train_donators)
    train_donations = sum(y_train_donators)
    train_min_profit = train_donations - (len(y_train)*var_cost)
    train_max_profit = train_donations - (train_donators*var_cost)
    print('%s train donators totaling $%.0f (min profit: $%.0f, max profit: $%.0f)' %\
          (train_donators, train_donations, train_min_profit, train_max_profit))
```

```
4812 train donators totalling $75113 (min profit: $10183, max profit: $71841)
```

Understanding The Effect of Irrelevant Features

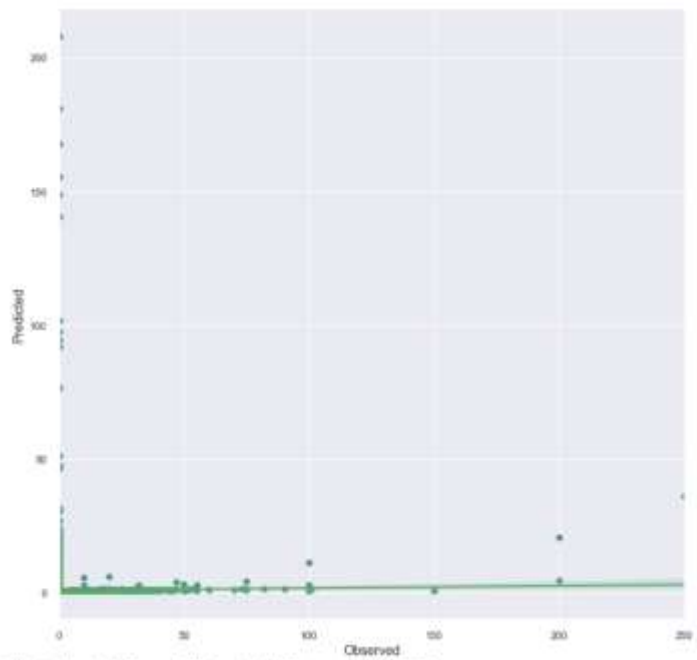
Creating a Base Model

```
[ ] rand = 9
    os.environ['PYTHONHASHSEED']=str(rand)
    np.random.seed(rand)
    orig_plt_params = plt.rcParams
    sns.set()
```

```
[ ] stime = timeit.default_timer()
    reg_mdl = xgb.XGBRFRegressor(max_depth=4, n_estimators=200, seed=rand)
    fitted_mdl = reg_mdl.fit(X_train, y_train)
    etime = timeit.default_timer()
    baseline_time = etime-stime
```

Evaluating the Model

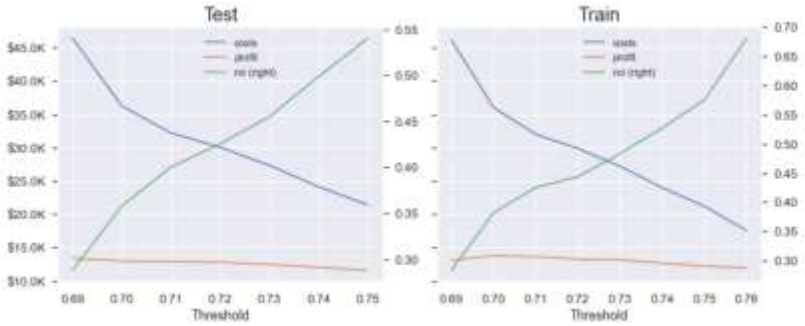
```
[ ] reg_mdls = {}
    reg_mdls['rf_4_all'] = mldatasets.evaluate_reg_mdl(fitted_mdl, X_train, X_test, y_train, y_test,\
                                        plot_regplot=True, ret_eval_dict=True)
    reg_mdls['rf_4_all']['speed'] = 1
    reg_mdls['rf_4_all']['depth'] = 4
    reg_mdls['rf_4_all']['fs'] = 'all'
```



RMSE_train: 4.3210 RMSE_test: 4.6359 r2: -0.1084

```
[ ] threshs = np.hstack([np.linspace(0.40,1,61), np.linspace(1.1,3,20),\
                         np.linspace(4,25,22)])
```

```
[ ] y_formatter = plt.FuncFormatter(lambda x, loc: "${:,}K".format(x/1000))

    profits_test = mldatasets.profits_by_thresh(y_test, reg_mdls['rf_4_all']['preds_test'],\
                                                threshs, var_costs=var_cost, min_profit=test_min_profit)
    profits_train = mldatasets.profits_by_thresh(y_train, reg_mdls['rf_4_all']['preds_train'],\
                                                 threshs, var_costs=var_cost, min_profit=train_min_profit)
    reg_mdls['rf_4_all']['max_profit_train'] = profits_train.profit.max()
    reg_mdls['rf_4_all']['max_profit_test'] = profits_test.profit.max()
    reg_mdls['rf_4_all']['max_roi'] = profits_test.roi.max()
    reg_mdls['rf_4_all']['min_costs'] = profits_test.costs.min()
    reg_mdls['rf_4_all']['profits_train'] = profits_train
    reg_mdls['rf_4_all']['profits_test'] = profits_test

    mldatasets.compare_df_plots(profits_test[['costs', 'profit', 'roi']],\
                                profits_train[['costs', 'profit', 'roi']],\
                                'Test', 'Train', x_label='Threshold',\
                                y_formatter=y_formatter, plot_args={'secondary_y':'roi'})
```



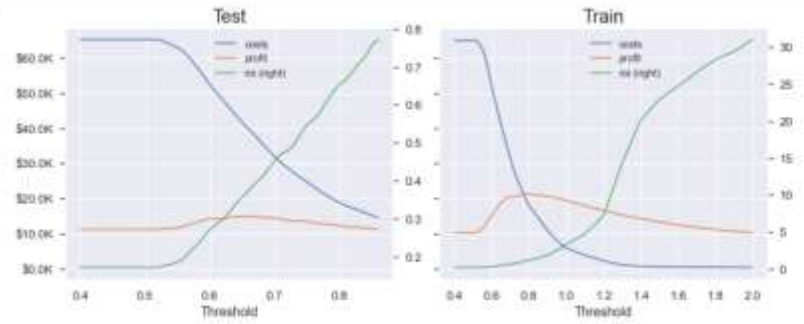
Training Base Model at Different Max Depths

```
[ ]  for depth in tqdm(range(5, 13)):
         mdlname = 'rf_'+str(depth)+'_all'
         stime = timeit.default_timer()
         reg_mdl = xgb.XGBRFRegressor(max_depth=depth, n_estimators=200, seed=rand)
         fitted_mdl = reg_mdl.fit(X_train, y_train)
         etime = timeit.default_timer()
         reg_mdls[mdlname] = mldatasets.evaluate_reg_mdl(fitted_mdl, X_train, X_test, y_train, y_test,\
                                                         plot_regplot=False, show_summary=False, ret_eval_dict=True)
         reg_mdls[mdlname]['speed'] = (etime-stime)/baseline_time
         reg_mdls[mdlname]['depth'] = depth
         reg_mdls[mdlname]['fs'] = 'all'
         profits_test = mldatasets.profits_by_thresh(y_test, reg_mdls[mdlname]['preds_test'],\
                                         threshs, var_costs=var_cost, min_profit=test_min_profit)
         profits_train = mldatasets.profits_by_thresh(y_train, reg_mdls[mdlname]['preds_train'],\
                                         threshs, var_costs=var_cost, min_profit=train_min_profit)
         reg_mdls[mdlname]['max_profit_train'] = profits_train.profit.max()
         reg_mdls[mdlname]['max_profit_test'] = profits_test.profit.max()
         reg_mdls[mdlname]['max_roi'] = profits_test.roi.max()
         reg_mdls[mdlname]['min_costs'] = profits_test.costs.min()
         reg_mdls[mdlname]['profits_train'] = profits_train
         reg_mdls[mdlname]['profits_test'] = profits_test
         reg_mdls[mdlname]['total_feat'] = reg_mdls[mdlname]['fitted'].feature_importances_.shape[0]
         reg_mdls[mdlname]['num_feat'] = sum(reg_mdls[mdlname]['fitted'].feature_importances_ > 0)
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=8.0), HTML(value='')))
```

```
[ ]  mldatasets.compare_df_plots(profits_test[['costs', 'profit', 'roi']],\
                                 profits_train[['costs', 'profit', 'roi']],\
                                 'Test', 'Train', x_label='Threshold',\
                                 y_formatter=y_formatter, plot_args={'secondary_y':'roi'})
```

```
[ ]
```




```
[ ]  reg_metrics_df = pd.DataFrame.from_dict(reg_mdls, 'index')\
                        [['depth', 'fs', 'rmse_train', 'rmse_test', 'max_profit_train',\
                          'max_profit_test', 'max_roi', 'min_costs', 'speed', 'num_feat']]
     with pd.option_context('display.precision', 2):
         html = reg_metrics_df.sort_values(by='depth', ascending=False).style.\
             background_gradient(cmap='plasma', low=0.1, high=1, subset=['rmse_train', 'rmse_test']).\
             background_gradient(cmap='viridis', low=1, high=0.1, subset=['max_profit_train', 'max_profit_test'])
     html
```

	depth	fs	rmse_train	rmse_test	max_profit_train	max_profit_test	max_roi	min_costs	speed	num_feat
rf_12_all	12	all	3.94	4.69	21521.98	14932.84	0.77	14532.28	2.89	415
rf_11_all	11	all	3.99	4.69	19904.00	15141.86	0.76	14928.04	2.73	398
rf_10_all	10	all	4.05	4.68	18603.92	14987.06	0.78	14396.28	2.43	383
rf_9_all	9	all	4.10	4.68	17453.14	14777.74	0.80	13997.12	2.19	346
rf_8_all	8	all	4.14	4.67	16439.72	14563.04	0.73	15308.84	1.94	315
rf_7_all	7	all	4.18	4.66	15435.32	14187.62	0.66	17164.56	1.71	277
rf_6_all	6	all	4.23	4.65	14651.12	13845.27	0.59	19305.20	1.41	240
rf_5_all	5	all	4.27	4.64	14242.32	13752.13	0.59	19199.12	1.22	201
rf_4_all	4	all	4.32	4.64	13518.60	13261.88	0.53	22392.40	1.00	160

Reviewing Filter-Based Feature Selection Methods

Basic Filter-Based Methods

Constant Features with Variance Threshold

```
num_cols_1 = X_train.select_dtypes([np.number]).columns
cat_cols_1 = X_train.select_dtypes([np.bool, np.object]).columns

num_const = VarianceThreshold(threshold=0)
num_const.fit(X_train[num_cols_1])

num_const_cols = list(set(X_train[num_cols_1].columns) - set(num_cols_1[num_const.get_support()]))
```

```
CPU times: user 575 ms, sys: 99 ms, total: 674 ms
Wall time: 672 ms
```

```
cat_const_cols = X_train[cat_cols_1].nunique()[lambda x: x<2].index.tolist()
all_const_cols = num_const_cols + cat_const_cols
print(all_const_cols)
```

```
['RFA_2R', 'REC_2', 'SOLP4']
```

Quasi-Constant Features with Value-Counts

```
thresh = 0.999
quasi_const_cols = []
num_rows = X_train.shape[0]
for col in tqdm(X_train.columns):
    top_val = (X_train[col].value_counts() /\
               num_rows).sort_values(ascending=False).values[0]

    if top_val >= thresh:
        quasi_const_cols.append(col)
```

Duplicate Features

```
X_train_transposed = X_train.T
dup_cols = X_train_transposed[X_train_transposed.duplicated()].index.tolist()
print(dup_cols)
```

```
['FREQ_2', 'RECINHSX']
```

Remove Unnecessary Features

```
X_train_orig = X_train.copy()
X_test_orig = X_test.copy()
drop_cols = quasi_const_cols + dup_cols
X_train.drop(labels=drop_cols, axis=1, inplace=True)
X_test.drop(labels=drop_cols, axis=1, inplace=True)
```

Correlation Filter-Based Methods

```
corrs = X_train.corr(method='spearman')
print(corrs.shape)
```

```
(428, 428)
CPU times: user 3min 21s, sys: 1.06 s, total: 3min 22s
Wall time: 3min 21s
```

```
extcorr_cols = (abs(corrs) > 0.99).sum(axis=1)[lambda x: x>1].index.tolist()
print(extcorr_cols)
uncorr_cols = (abs(corrs) > 0.15).sum(axis=1)[lambda x: x==1].index.tolist()
print(uncorr_cols)
```

```
['MAJOR', 'HHAGE1', 'HHAGE3', 'HHN3', 'HHP1', 'HV1', 'HV2', 'MDMAUD_R', 'MDMAUD_F', 'MDMAUD_A']
['TCODE', 'MAILCODE', 'NOEXCH', 'CHILD03', 'CHILD07', 'CHILD12', 'CHILD18', 'HC15', 'MAXADATE']
```

```
corr_cols = X_train.columns[~X_train.columns.isin(uncorr_cols)].tolist()
print(len(corr_cols))
```

Ranking Filter-Based Methods

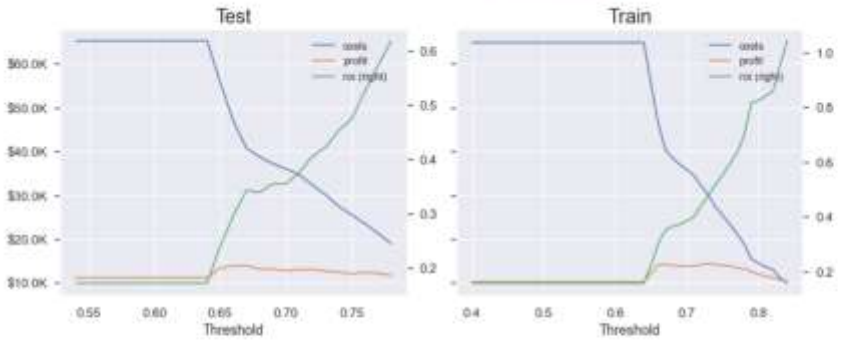
```
[ ] y_train_class = np.where(y_train > 0.68, 1, 0)
```

```
[ ] mic_selection = SelectKBest(mutual_info_classif, k=160).fit(X_train, y_train_class)
    mic_cols = X_train.columns[mic_selection.get_support()].tolist()
    print(len(mic_cols))
```

```
160
CPU times: user 4min 54s, sys: 2.48 s, total: 4min 57s
Wall time: 4min 57s
```

```
[ ] mdlname = 'rf_5_f-mic'
    stime = timeit.default_timer()
    reg_mdl = xgb.XGBRFRegressor(max_depth=5, n_estimators=200, seed=rand)
    fitted_mdl = reg_mdl.fit(X_train[mic_cols], y_train)
    etime = timeit.default_timer()
    reg_mdls[mdlname] = mldatasets.evaluate_reg_mdl(fitted_mdl, X_train[mic_cols], X_test[mic_cols], y_train, y_test,\
                                                    plot_regplot=False, show_summary=False, ret_eval_dict=True)
    reg_mdls[mdlname]['speed'] = (etime-stime)/baseline_time
    reg_mdls[mdlname]['depth'] = 5
    reg_mdls[mdlname]['fs'] = 'f-mic'
    profits_test = mldatasets.profits_by_thresh(y_test, reg_mdls[mdlname]['preds_test'],\
                                thresholds, var_costs=var_cost, min_profit=test_min_profit)
    profits_train = mldatasets.profits_by_thresh(y_train, reg_mdls[mdlname]['preds_train'],\
                                thresholds, var_costs=var_cost, min_profit=train_min_profit)
    reg_mdls[mdlname]['max_profit_train'] = profits_train.profit.max()
    reg_mdls[mdlname]['max_profit_test'] = profits_test.profit.max()
    reg_mdls[mdlname]['max_roi'] = profits_test.roi.max()
    reg_mdls[mdlname]['min_costs'] = profits_test.costs.min()
    reg_mdls[mdlname]['profits_train'] = profits_train
    reg_mdls[mdlname]['profits_test'] = profits_test
    reg_mdls[mdlname]['total_feat'] = reg_mdls[mdlname]['fitted'].feature_importances_.shape[0]
    reg_mdls[mdlname]['num_feat'] = sum(reg_mdls[mdlname]['fitted'].feature_importances_ > 0)
```

```
[ ] mldatasets.compare_df_plots(profits_test[['costs', 'profit', 'roi']],\
                                profits_train[['costs', 'profit', 'roi']],\
                                'Test', 'Train', x_label='Threshold',\
                                y_formatter=y_formatter, plot_args={'secondary_y':'roi'})
```



Comparing Filter-based Methods

```
[ ] reg_metrics_df = pd.DataFrame.from_dict(reg_mdls, 'index')\
                      [['depth', 'fs', 'rmse_train', 'rmse_test', 'max_profit_train',\
                        'max_profit_test', 'max_roi', 'min_costs', 'speed', 'total_feat', 'num_feat']]
    with pd.option_context('display.precision', 2):
        html = reg_metrics_df.sort_values(by='max_profit_test', ascending=False).style.\
            background_gradient(cmap='plasma', low=0.3, high=1, subset=['rmse_train', 'rmse_test']).\
            background_gradient(cmap='viridis', low=1, high=0.3, subset=['max_profit_train', 'max_profit_test'])
    html
```

```
html
```

	depth	fs	rmse_train	rmse_test	max_profit_train	max_profit_test	max_roi	min_costs	speed	total_feat	num_feat
rf_11_all	11	all	3.99	4.69	19904.00	15141.86	0.76	14928.04	2.73	435	398
rf_10_all	10	all	4.05	4.68	18603.92	14987.06	0.78	14396.28	2.43	435	383
rf_12_all	12	all	3.94	4.69	21521.98	14932.84	0.77	14532.28	2.89	435	415
rf_11_f-corr	11	f-corr	3.98	4.67	19923.84	14894.94	0.77	14592.80	2.47	419	404
rf_9_all	9	all	4.10	4.68	17453.14	14777.74	0.80	13997.12	2.19	435	346
rf_8_all	8	all	4.14	4.67	16439.72	14563.04	0.73	15308.84	1.94	435	315
rf_5_f-mic	5	f-mic	4.31	4.57	14983.34	14481.39	0.62	18971.32	0.39	160	103
rf_7_all	7	all	4.18	4.66	15435.32	14187.62	0.66	17164.56	1.71	435	277
rf_6_all	6	all	4.23	4.65	14651.12	13845.27	0.59	19305.20	1.41	435	240
rf_5_all	5	all	4.27	4.64	14242.32	13752.13	0.59	19199.12	1.22	435	201
rf_4_all	4	all	4.32	4.64	13715.90	13261.88	0.53	22392.40	1.00	435	160

- Exploring Embedded Feature Selection Methods

- Lasso

```
[ ] lasso_selection = SelectFromModel(LassoCV(n_jobs=-1, random_state=rand))
    lasso_selection.fit(X_train, y_train)
    lasso_cols = X_train.columns[lasso_selection.get_support()].tolist()
    print(len(lasso_cols))
    print(lasso_cols)
```

```
7
['ODATEDW', 'TCODE', 'POP901', 'POP902', 'HV2', 'RAMNTALL', 'MAXRDATE']
```

- Lasso Lars

```
[ ] llars_selection = SelectFromModel(LassoLarsCV(n_jobs=-1))
    llars_selection.fit(X_train, y_train)
    llars_cols = X_train.columns[llars_selection.get_support()].tolist()
```

```
[ ] reg_metrics_df = pd.DataFrame.from_dict(reg_mdls, 'index')\
                    [['depth', 'fs', 'rmse_train', 'rmse_test', 'max_profit_train',\
                      'max_profit_test', 'max_roi', 'min_costs', 'speed', 'total_feat', 'num_feat']]
    with pd.option_context('display.precision', 2):
        html = reg_metrics_df.sort_values(by='max_profit_test', ascending=False).style.\
            background_gradient(cmap='plasma', low=0.3, high=1, subset=['rmse_train', 'rmse_test']).\
            background_gradient(cmap='viridis', low=1, high=0.3, subset=['max_profit_train', 'max_profit_test'])
    html
```

	depth	fs	rmse_train	rmse_test	max_profit_train	max_profit_test	max_roi	min_costs	speed	total_feat	num_feat
rf_11_all	11	all	3.99	4.69	19904.00	15141.86	0.76	14928.04	2.73	435	398
rf_10_all	10	all	4.05	4.68	18603.92	14987.06	0.78	14396.28	2.43	435	383
rf_12_all	12	all	3.94	4.69	21521.98	14932.84	0.77	14532.28	2.89	435	415
rf_11_f-corr	11	f-corr	3.98	4.67	19923.84	14894.94	0.77	14592.80	2.47	419	404
rf_9_all	9	all	4.10	4.68	17453.14	14777.74	0.80	13997.12	2.19	435	346
rf_5_e-llarsic	5	e-llarsic	4.28	4.45	[illegible]	14768.37	0.56	20441.48	0.32	111	87
rf_8_all	8	all	4.14	4.67	16439.72	14563.04	0.73	15308.84	1.94	435	315
rf_5_f-mic	5	f-mic	4.31	4.57	14983.34	14481.39	0.62	18971.32	0.39	160	103
rf_6_e-logl2	6	e-logl2	4.28	4.60	[illegible]	14189.90	0.67	16904.12	0.32	87	84
rf_7_all	7	all	4.18	4.66	15435.32	14187.62	0.66	17164.56	1.71	435	277
rf_6_all	6	all	4.23	4.65	14651.12	13845.27	0.59	19305.20	1.41	435	240
rf_5_all	5	all	4.27	4.64	14242.32	13752.13	0.59	19199.12	1.22	435	201
rf_4_e-llars	4	e-llars	4.36	4.45	[illegible]	[illegible]	0.52	22906.48	0.06	8	8
rf_4_all	4	all	4.32	4.64	13715.90	13261.88	0.53	22392.40	1.00	435	160
rf_3_e-lasso	3	e-lasso	4.46	4.49	[illegible]	[illegible]	0.51	22248.82	0.06	7	7

- Genetic Algorithms

```
[ ] ga_rf = GeneticSelectionCV(RandomForestRegressor(random_state=rand, max_depth=3),\
                               cv=3, scoring='neg_root_mean_squared_error', max_features=[illegible],\
                               crossover_independent_proba=[illegible], n_gen_no_change=[illegible],\
                               mutation_independent_proba=[illegible], n_jobs=-1, verbose=2)
    ga_rf = ga_rf.fit(X_train[top_cols], y_train)
    ga_rf_cols = np.array(top_cols)[ga_rf.support_].tolist()
```

```
Selecting features with genetic algorithm.
gen     nevals  avg                             std                             min                             max
0       300     [-5.40823336 45.95      ]       [ 0.07125515 27.40443099]       [-5.63924509  1.        ]       [-5.20095401 90.        ]
1       168     [-105.30715505   35.35333333]   [394.45400906  22.64574218]     [-1.e+04  1.e+00]               [-5.20193002 94.        ]
2       181     [-5.34085156 26.85333333]       [ 0.04440001 16.3374470 ]       [-5.62536741  1.        ]       [-5.2727256 80.        ]
3       198     [-5.33136005 22.12333333]       [ 0.04187036 13.65777784]       [-5.58671499  1.        ]       [-5.27094437 72.        ]
4       204     [-5.3267442 20.71     ]         [ 0.04401984 11.83832336]       [-5.57742955  1.        ]       [-5.27128264 66.        ]
5       185     [-5.3172738  20.35333333]       [ 0.03794402 10.35962054]       [-5.53270449  1.        ]       [-5.24907367 55.        ]
[illegible]
```

Evaluating all Feature Selected Models

```
fsnames = ['w-sfs-lda', 'w-sbs-et', 'h-rfe-lda','h-rfe-rf', 'a-pca', 'a-shap', 'a-ga-rf']
fscols = [sfs_lda_cols, sbs_et_cols, rfe_lda_cols, rfe_rf_cols, pca_cols, shap_cols, ga_rf_cols]
depths = [5, 6, 6, 6, 6, 6, 5]
```

```
for i, fsname in tqdm(enumerate(fsnames), total=len(fsnames)):
    depth = depths[i]
    cols = fscols[i]
    mdlname = 'rf_'+str(depth)+'_'+fsname
    stime = time.perf_counter()
    reg_mdl = xgb.XGBRFRegressor(max_depth=depth, n_estimators=200, seed=rand)
    fitted_mdl = reg_mdl.fit(X_train[cols], y_train)
    etime = time.perf_counter()
    reg_mdls[mdlname] = mldatasets.evaluate_reg_mdl(fitted_mdl, X_train[cols],\
                                                    X_test[cols], y_train, y_test, plot_regplot=False,\
                                                    show_summary=False, ret_eval_dict=True)
    reg_mdls[mdlname]['speed'] = (etime-stime)/baseline_time
    reg_mdls[mdlname]['depth'] = depth
    reg_mdls[mdlname]['fs'] = fsname
    profits_test = mldatasets.profits_by_thresh(y_test, reg_mdls[mdlname]['preds_test'],\
                                                thresh=, var_costs=var_cost, min_profit=test_min_profit)
    profits_train = mldatasets.profits_by_thresh(y_train, reg_mdls[mdlname]['preds_train'],\
                                                 thresh=, var_costs=var_cost, min_profit=train_min_profit)
    reg_mdls[mdlname]['max_profit_train'] = profits_train.profit.max()
    reg_mdls[mdlname]['max_profit_test'] = profits_test.profit.max()
    reg_mdls[mdlname]['max_roi'] = profits_test.roi.max()
    reg_mdls[mdlname]['min_costs'] = profits_test.costs.min()
    reg_mdls[mdlname]['profits_train'] = profits_train
    reg_mdls[mdlname]['profits_test'] = profits_test
    reg_mdls[mdlname]['total_feat'] = reg_mdls[mdlname]['fitted'].feature_importances_.shape[0]
    reg_mdls[mdlname]['num_feat'] = sum(reg_mdls[mdlname]['fitted'].feature_importances_ > 0)
```

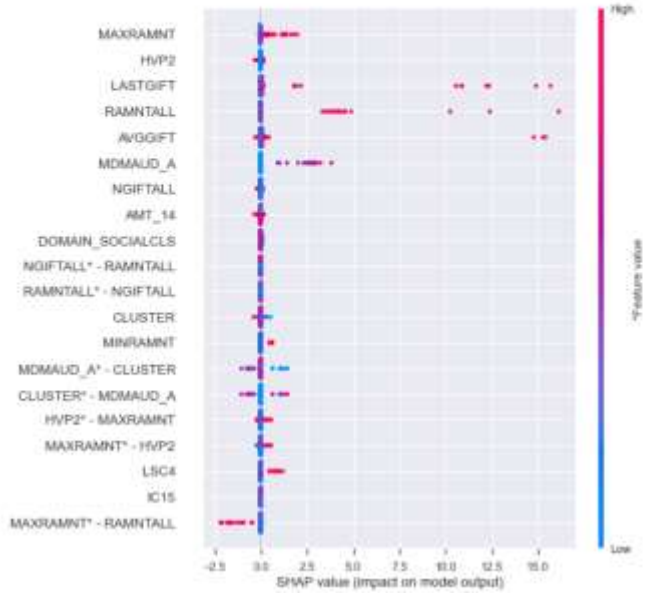
```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=5.0), HTML(value='')))
```

```
reg_metrics_df = pd.DataFrame.from_dict(reg_mdls, 'index')\
                    [['depth', 'fs', 'rmse_train', 'rmse_test', 'max_profit_train',\
                      'max_profit_test', 'max_roi', 'min_costs', 'speed', 'total_feat', 'num_feat']]
reg_metrics_df = reg_metrics_df[reg_metrics_df.depth < 7]
with pd.option_context('display.precision', 2):
    html = reg_metrics_df.sort_values(by='max_profit_test', ascending=False).style.\
        background_gradient(cmap='plasma', low=0.3, high=1, subset=['rmse_train', 'rmse_test']).\
        background_gradient(cmap='viridis', low=1, high=0.1, subset=['max_profit_train', 'max_profit_test'])
html
```

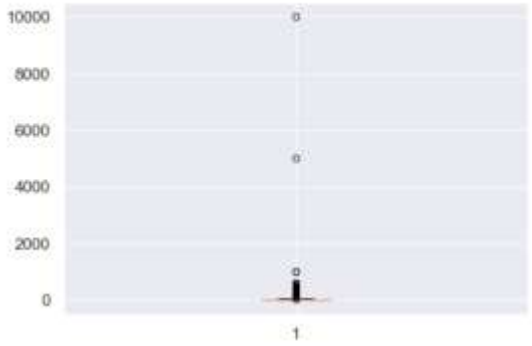
	depth	fs	rmse_train	rmse_test	max_profit_train	max_profit_test	max_roi	min_costs	speed	total_feat	num_feat
rf_5_e-liersic	5	e-liersic	4.28	4.45	95165.46	14799.37	0.56	20441.46	0.32	111	87
rf_5_f-mic	5	f-mic	4.31	4.57	94963.34	14481.39	0.62	19871.32	0.38	160	103
rf_6_h-rfe-lda	6	h-rfe-lda	4.25	4.48	95339.72	14351.74	0.71	15624.26	0.61	165	129
rf_6_a-shap	6	a-shap	4.23	4.52	95263.60	14282.28	0.61	19767.32	0.56	150	126
rf_5_a-ga-rf	5	a-ga-rf	4.28	4.45	94274.52	14220.63	0.69	13237.56	0.57	83	63
rf_6_a-log2	6	a-log2	4.28	4.48	95265.44	14199.80	0.67	16964.12	0.32	87	84
rf_6_all	6	all	4.23	4.69	94951.12	13949.27	0.58	19305.20	1.41	405	240
rf_5_w-sfs-lda	5	w-sfs-lda	4.43	4.63	94377.15	13801.99	0.61	22566.95	0.11	27	27
rf_5_all	5	all	4.27	4.64	94542.32	13752.13	0.59	19199.12	1.22	405	201
rf_4_e-liars	4	e-liars	4.36	4.46	94014.91	13633.59	0.52	22666.46	0.08	8	8
rf_6_a-pca	6	a-pca	4.30	4.46	95263.54	13351.97	0.47	23601.52	0.54	150	126
rf_6_h-rfe-rf	6	h-rfe-rf	4.48	4.78	93202.81	13347.55	0.41	26566.04	0.08	1	1
rf_4_all	4	all	4.12	4.64	93715.66	13301.84	0.53	22362.40	1.00	405	160
rf_6_w-sbs-et	6	w-sbs-et	4.34	4.53	94522.49	13118.17	0.71	14711.68	0.45	135	123
rf_3_e-lasso	3	e-lasso	4.46	4.49	94165.94	13083.20	0.51	22243.92	0.05	7	7

Considering Feature Engineering

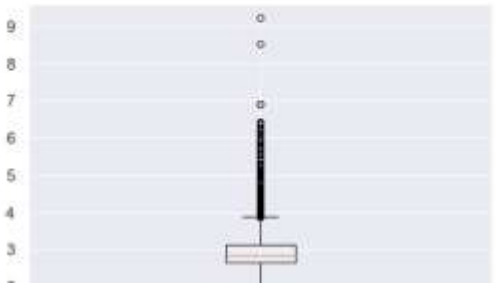
```
[ ] fitted_rf_mdl = reg_mdls['rf_5_e-llarsic']['fitted']
    shap_rf_explainer = shap.TreeExplainer(fitted_rf_mdl)
    shap_rf_interact_values =\
            shap_rf_explainer.shap_interaction_values(X_test.iloc[sample_test_idx][llarsic_cols])
    shap.summary_plot(shap_rf_interact_values, X_test.iloc[sample_test_idx][llarsic_cols],\
                      plot_type="compact_dot", sort=True)
```



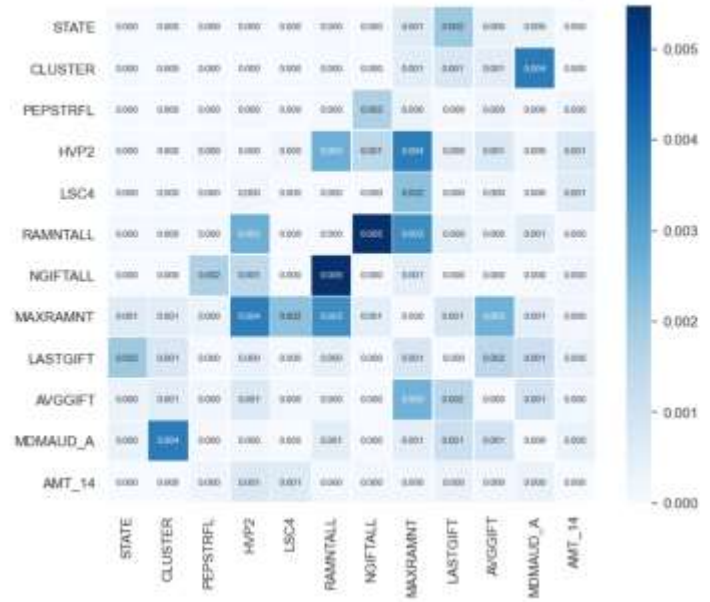

```
[ ] plt.boxplot(X_test.MAXRAMNT)
    plt.show()
```



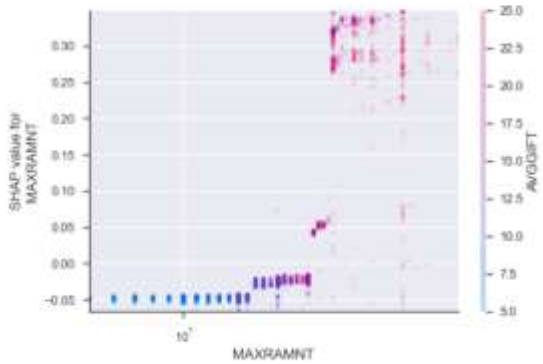

```
[ ] plt.boxplot(np.log(X_test.MAXRAMNT))
    plt.show()
```



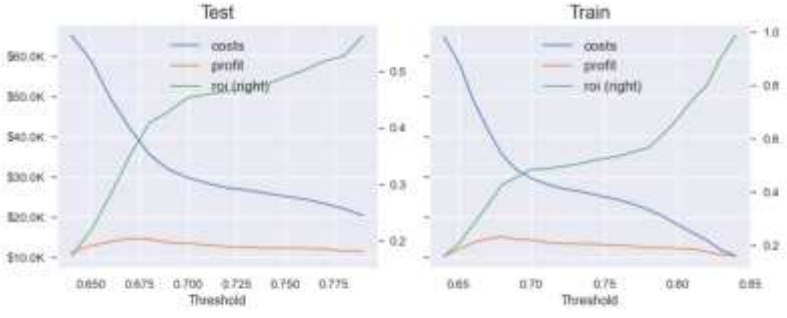
```
[ ] sns.heatmap(shap_rf_interact_df, cmap='Blues', annot=True,\
                annot_kws={'size':10}, fmt='.3f', linewidths=.5)
    plt.show()
```




```
[ ] shap_rf_values = shap_rf_explainer.shap_values(X_test.iloc[sample_test_idx][llarsic_cols])
    maxramt_shap = shap_rf_values[:,llarsic_cols.index("MAXRAMNT")]
    shap.dependence_plot("MAXRAMNT", shap_rf_values, X_test.iloc[sample_test_idx][llarsic_cols],\
                         interaction_index="AVGGIFT", show=False, alpha=0.1)
    plt.xlim(xmin=np.percentile(X_test.MAXRAMNT, 1), xmax=np.percentile(X_test.MAXRAMNT, 99))
    plt.ylim(ymin=np.percentile(maxramt_shap, 1), ymax=np.percentile(maxramt_shap, 99))
    plt.xscale('log')
    plt.show()
```




```
[ ] profits_test = reg_mdls['rf_5_e-llarsic']['profits_test']
    profits_train = reg_mdls['rf_5_e-llarsic']['profits_train']
    mldatasets.compare_df_plots(profits_test[['costs', 'profit', 'roi']],\
                                profits_train[['costs', 'profit', 'roi']],\
                                'Test', 'Train', x_label='Threshold',\
                                y_formatter=y_formatter, plot_args={'secondary_y':'roi'})
```



Chapter 11 (Credit card defaults):

```
!pip install --upgrade pandas numpy tqdm scikit-learn lightgbm xgboost networkx pydot matplotlib seaborn
```

```
Collecting pandas
  Downloading pandas-1.5.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.2/12.2 MB 43.2 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (1.21.6)
Collecting numpy
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 28.0 MB/s eta 0:00:00
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (4.64.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.8/dist-packages (1.0.2)
Collecting scikit-learn
  Downloading scikit_learn-1.2.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (9.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.8/9.8 MB 30.4 MB/s eta 0:00:00
Requirement already satisfied: lightgbm in /usr/local/lib/python3.8/dist-packages (2.2.3)
Collecting lightgbm
  Downloading lightgbm-3.3.5-py3-none-manylinux1_x86_64.whl (2.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 23.4 MB/s eta 0:00:00
Requirement already satisfied: xgboost in /usr/local/lib/python3.8/dist-packages (0.90)
Collecting xgboost
  Downloading xgboost-1.7.3-py3-none-manylinux2014_x86_64.whl (193.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 193.6/193.6 MB 3.3 MB/s eta 0:00:00
Requirement already satisfied: networkx in /usr/local/lib/python3.8/dist-packages (3.0)
Requirement already satisfied: pydot in /usr/local/lib/python3.8/dist-packages (1.3.0)
Collecting pydot
  Downloading pydot-1.4.2-py2.py3-none-any.whl (21 kB)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (3.2.2)
Collecting matplotlib
  Downloading matplotlib-3.6.3-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (9.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.4/9.4 MB 24.8 MB/s eta 0:00:00
Requirement already satisfied: seaborn in /usr/local/lib/python3.8/dist-packages (0.11.2)
Collecting seaborn
  Downloading seaborn-0.12.2-py3-none-any.whl (293 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 293.3/293.3 KB 8.7 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (3.1.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: wheel in /usr/local/lib/python3.8/dist-packages (from lightgbm) (0.38.4)
Requirement already satisfied: pyparsing>=2.1.4 in /usr/local/lib/python3.8/dist-packages (from pydot) (3.0.9)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (7.1.2)
Collecting fonttools>=4.22.0
  Downloading fonttools-4.38.0-py3-none-any.whl (965 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 965.4/965.4 KB 32.4 MB/s eta 0:00:00
```

Loading the Libraries

```
[ ]  import math
     import os
     import warnings
     warnings.filterwarnings("ignore")
     import machine_learning_datasets as mldatasets
     import pandas as pd
     import numpy as np
     from tqdm.notebook import tqdm

     from sklearn import model_selection, tree, metrics
     import lightgbm as lgb
     import xgboost as xgb

     from aif360.datasets import BinaryLabelDataset
     from aif360.metrics import BinaryLabelDatasetMetric
     from aif360.metrics import ClassificationMetric
     from aif360.algorithms.preprocessing import Reweighing
     from aif360.algorithms.preprocessing import DisparateImpactRemover
     from aif360.algorithms.inprocessing import PrejudiceRemover
     from aif360.algorithms.inprocessing import GerryFairClassifier
     from aif360.algorithms.postprocessing.calibrated_eq_odds_postprocessing import CalibratedEqOddsPostprocessing
     from aif360.algorithms.postprocessing.eq_odds_postprocessing import EqOddsPostprocessing

     from econml.dr import LinearDRLearner
     import dowhy
     from dowhy import CausalModel

     import xai
     from networkx.drawing.nx_pydot import to_pydot
     from IPython.display import Image, display
     import matplotlib.pyplot as plt
     import seaborn as sns

     %matplotlib inline
```

Understanding and Preparing the Data

```
[ ] ccdefault_all_df = mldatasets.load("cc_default", prepare=True)
```

```
[ ] ccdefault_all_df.info()
```

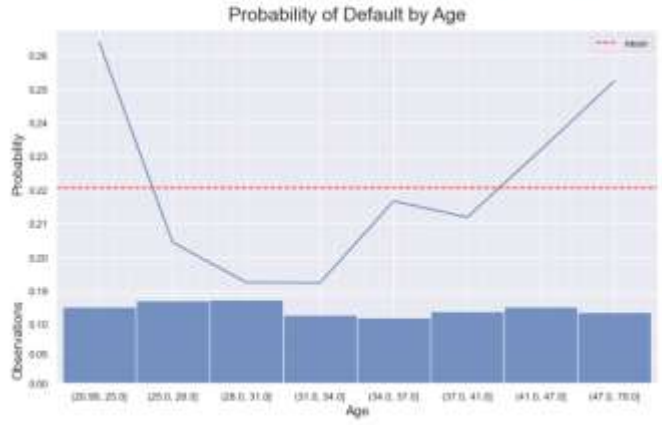
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30000 entries, 0 to 30000
Data columns (total 31 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   LI_LIMIT_CAT    30000 non-null  int8
 1   EDUCATION       30000 non-null  int8
 2   MARITAL_STATUS  30000 non-null  int8
 3   GENDER          30000 non-null  int8
 4   AGE_GROUP       30000 non-null  int8
 5   pay_status_1    30000 non-null  int8
 6   pay_status_2    30000 non-null  int8
 7   pay_status_3    30000 non-null  int8
 8   pay_status_4    30000 non-null  int8
 9   pay_status_5    30000 non-null  int8
 10  pay_status_6    30000 non-null  int8
 11  paid_pct_1      30000 non-null  float64
 12  paid_pct_2      30000 non-null  float64
 13  paid_pct_3      30000 non-null  float64
 14  paid_pct_4      30000 non-null  float64
 15  paid_pct_5      30000 non-null  float64
 16  paid_pct_6      30000 non-null  float64
 17  bill_over_limit 30000 non-null  float64
 18  IS_DEFAULT      30000 non-null  int8
 19  _AGE            30000 non-null  int16
 20  _speed          30000 non-null  int32
 21  _tpm            30000 non-null  int16
 22  _gpm            30000 non-null  int16
 23  _RETAIL         30000 non-null  int8
 24  _DRINK          30000 non-null  int8
```

Visualizing Dataset Bias

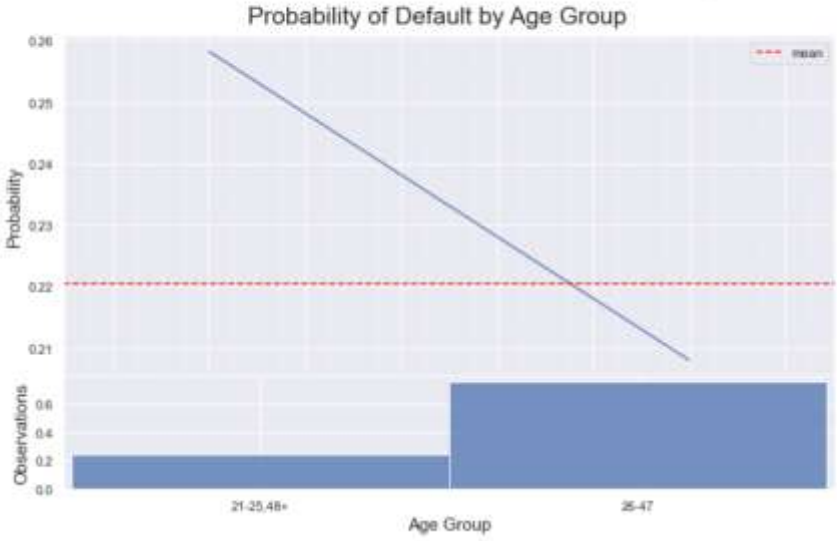
```
[ ] ccdefault_bias_df[ccdefault_bias_df.IS_DEFAULT==1].GENDER.\
              value_counts()/ccdefault_bias_df.GENDER.value_counts()
```

```
2    0.208529
1    0.241633
Name: GENDER, dtype: float64
```

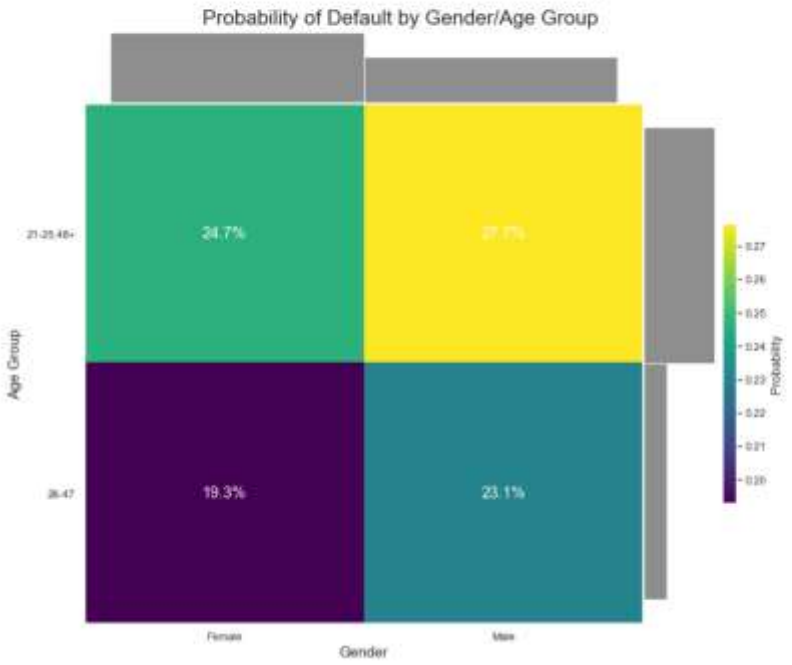
```
[ ] mldatasets.plot_prob_progression(ccdefault_bias_df._AGE, ccdefault_bias_df.IS_DEFAULT,\
                                     x_intervals=8, use_quartiles=True, xlabel='Age',\
                                     title='Probability of Default by Age', mean_line=True)
```




```
[ ] mldatasets.plot_prob_progression(ccdefault_bias_df.AGE_GROUP.replace({0:'21-25,48+',1:'26-47'}),\
                                     ccdefault_bias_df.IS_DEFAULT, xlabel='Age Group',\
                                     title='Probability of Default by Age Group', mean_line=True)
```



```
[ ] wldatasets.plot_prob_contour_map(ccdefault_bias_df.GENDER.replace({1:'Male',2:'Female'}),\
                        ccdefault_bias_df.AGE_GROUP.replace({0:'21-25,48+',1:'26-47'}),\
                        ccdefault_bias_df.IS_DEFAULT, xlabel='Gender', ylabel='Age Group',\
                    title='Probability of Default by Gender/Age Group', plot_type='grid',\
                    annotate=True)
```



Quantifying Dataset Bias

```
[ ] cols_bias_1 =\
        ccdefault_all_df.columns[~ccdefault_all_df.columns.str.startswith('_')].tolist()
    cols_causal_1 =\
        ['AGE_GROUP','IS_DEFAULT'] +\
        ccdefault_all_df.columns[ccdefault_all_df.columns.str.startswith('_')].tolist()

    ccdefault_bias_df = ccdefault_bias_df[cols_bias_1]
    ccdefault_causal_df = ccdefault_causal_df[cols_causal_1]
```

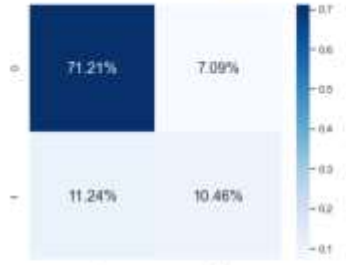
```
[ ] rand = 9
    os.environ['PYTHONHASHSEED']=str(rand)
    np.random.seed(rand)

    y = ccdefault_bias_df['IS_DEFAULT']
    X = ccdefault_bias_df.drop(['IS_DEFAULT'], axis=1).copy()
    X_train, X_test, y_train, y_test =\
        model_selection.train_test_split(X, y, test_size=0.25, random_state=rand)
```

```
[ ] train_ds = BinaryLabelDataset(df=X_train.join(y_train),\
                        label_names=['IS_DEFAULT'],\
                        protected_attribute_names=['AGE_GROUP', 'GENDER'],
                        favorable_label=0, unfavorable_label=1)
    test_ds = BinaryLabelDataset(df=X_test.join(y_test),\
                        label_names=['IS_DEFAULT'],\
                        protected_attribute_names=['AGE_GROUP', 'GENDER'],
                        favorable_label=0, unfavorable_label=1)
```

```
[ ] unprivileged_groups=[{'AGE_GROUP': 0}]
    privileged_groups=[{'AGE_GROUP': 1}]

    metrics_train_ds = BinaryLabelDatasetMetric(train_ds,\
                                unprivileged_groups=unprivileged_groups,\
                                privileged_groups=privileged_groups)
    metrics_test_ds = BinaryLabelDatasetMetric(test_ds,\
                                unprivileged_groups=unprivileged_groups,\
                                privileged_groups=privileged_groups)
```

Quantifying Model Bias

```
[ ] cls_mdls = {}

    lgb_params = {'learning_rate': 0.4, 'reg_alpha': 25, 'reg_lambda': 1,\
              'scale_pos_weight': 1.8}
    lgb_base_mdl = lgb.LGBMClassifier(random_seed=rand, max_depth=6,\
                                      num_leaves=33, **lgb_params)
    lgb_base_mdl.fit(X_train, y_train)
    cls_mdls['lgb_0_base'] =\
        mldatasets.evaluate_class_mdl(lgb_base_mdl,\
                X_train, X_test, y_train, y_test, plot_roc=False,\
                plot_conf_matrix=True, show_summary=True, ret_eval_dict=True)
```




```
Accuracy_train:  0.8255        Accuracy_test:   0.8167
Precision_test:  0.5961        Recall_test:     0.4828
ROC-AUC_test:    0.7901        F1_test:         0.5330        MCC_test: 0.4243
```

```
[ ] test_pred_ds = test_ds.copy(deepcopy=True)
    test_pred_ds.labels = cls_mdls['lgb_0_base']['preds_test'].reshape(-1,1)
    test_pred_ds.scores = cls_mdls['lgb_0_base']['probs_test'].reshape(-1,1)
    metrics_test_dict, metrics_test_cls =\
            mldatasets.compute_aif_metrics(test_ds, test_pred_ds,\
```

Mitigating Bias

Pre-processing Bias Mitigation Methods

Reweighting Method

```
[ ] reweighter = Reweighing(unprivileged_groups=unprivileged_groups,\
                            privileged_groups=privileged_groups)
    reweighter.fit(train_ds)
    train_rw_ds = reweighter.transform(train_ds)
```

```
[ ] metrics_train_rw_ds = BinaryLabelDatasetMetric(train_rw_ds,\
                                unprivileged_groups=unprivileged_groups,\
                                privileged_groups=privileged_groups)

    print('Statistical Parity Difference (SPD):\t\t\t%.4f' %\
          metrics_train_rw_ds.statistical_parity_difference())
    print('Disparate Impact (DI):\t\t\t\t%.4f' %\
          metrics_train_rw_ds.disparate_impact())
    print('Smoothed Empirical Differential Fairness (SEDF):\t%.4f' %\
          metrics_train_rw_ds.smoothed_empirical_differential_fairness())
```

```
Statistical Parity Difference (SPD):                -0.0000
Disparate Impact (DI):                              1.0000
Smoothed Empirical Differential Fairness (SEDF):    0.1942
```

```
[ ] np.abs(train_ds.instance_weights.mean() - train_rw_ds.instance_weights.mean()) < 1e-6
```

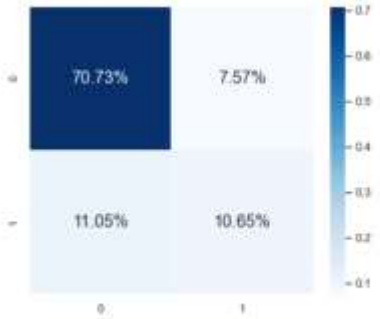
```
True
```

```
[ ] lgb_rw_mdl = lgb.LGBMClassifier(random_seed=rand, max_depth=6,\
                                    num_leaves=33, **lgb_params)
    lgb_rw_mdl.fit(X_train, y_train,
                   sample_weight=train_rw_ds.instance_weights)
```

```
[ ]                                num_leaves=33, **lgb_params)
   lgb_rw_mdl.fit(X_train, y_train,
                  sample_weight=train_rw_ds.instance_weights)
```

```
LGBMClassifier(learning_rate=0.4, max_depth=6, num_leaves=33, random_seed=9,
               reg_alpha=21, reg_lambda=1, scale_pos_weight=1.8)
```

```
[ ] cls_mdls['lgb_1_rw'] =\
        mldatasets.evaluate_class_mdl(lgb_rw_mdl,\
                train_rw_ds.features, X_test, train_rw_ds.labels, y_test,\
                plot_roc=False, plot_conf_matrix=True, show_summary=True,\
                ret_eval_dict=True)

    test_pred_rw_ds = test_ds.copy(deepcopy=True)
    test_pred_rw_ds.labels = cls_mdls['lgb_1_rw']['preds_test'].reshape(-1,1)
    test_pred_rw_ds.scores = cls_mdls['lgb_1_rw']['probs_test'].reshape(-1,1)
    metrics_test_rw_dict, _ =\
            mldatasets.compute_aif_metrics(test_ds, test_pred_rw_ds,\
                                unprivileged_groups=unprivileged_groups,\
                                privileged_groups=privileged_groups)
    cls_mdls['lgb_1_rw'].update(metrics_test_rw_dict)
```



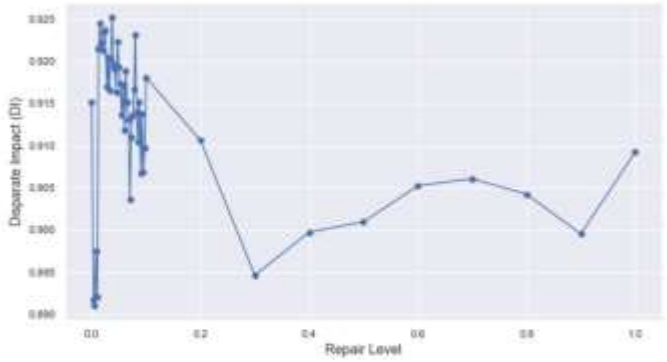

```
Accuracy_train:  0.8240          Accuracy_test:   0.8138
Precision_test:  0.5847          Recall_test:     0.4908
ROC-AUC_test:    0.7886          F1_test:         0.5337          MCC_test: 0.4210
```

Disparate Impact Remover Method

```
[ ] protected_index = train_ds.feature_names.index('AGE_GROUP')
```

```
[ ] di = np.array([])
    train_dir_ds = None
    test_dir_ds = None
    lgb_dir_mdl = None
    X_train_dir = None
    X_test_dir = None

    levels = np.hstack([np.linspace(0., 0.1, 41), np.linspace(0.2, 1, 9)])

    for level in tqdm(levels):
        di_remover = DisparateImpactRemover(repair_level=level)
        train_dir_ds_i = di_remover.fit_transform(train_ds)
        test_dir_ds_i = di_remover.fit_transform(test_ds)

        X_train_dir_i = np.delete(train_dir_ds_i.features, protected_index, axis=1)
        X_test_dir_i = np.delete(test_dir_ds_i.features, protected_index, axis=1)

        lgb_dir_mdl_i = lgb.LGBMClassifier(random_seed=rand, max_depth=5,\
                                num_leaves=33, **lgb_params)
        lgb_dir_mdl_i.fit(X_train_dir_i, train_dir_ds_i.labels)

        test_dir_ds_pred_i = test_dir_ds_i.copy()
        test_dir_ds_pred_i.labels = lgb_dir_mdl_i.predict(X_test_dir_i)
        metrics_test_dir_ds = BinaryLabelDatasetMetric(test_dir_ds_pred_i,\
                                    unprivileged_groups=unprivileged_groups,\
                                    privileged_groups=privileged_groups)
        di_i = metrics_test_dir_ds.disparate_impact()

        if (di.shape[0]==0) or (np.min(np.abs(di-1)) >= abs(di_i-1)):
            print(abs(di_i-1))
            train_dir_ds = train_dir_ds_i
            test_dir_ds = test_dir_ds_i
            X_train_dir = X_train_dir_i
            X_test_dir = X_test_dir_i
            lgb_dir_mdl = lgb_dir_mdl_i
```

```
[ ] sns.set()
    plt.figure(figsize=(11,6))
    plt.plot(levels, di, marker='o')
    plt.ylabel('Disparate Impact (DI)', fontsize=14)
    plt.xlabel('Repair Level', fontsize=14)
    #plt.xscale('log')
    plt.show()
```




```
[ ] cls_mdls['lgb_1_dir'] =\
        mldatasets.evaluate_class_mdl(lgb_dir_mdl,\
                X_train_dir, X_test_dir, train_dir_ds.labels, test_dir_ds.labels,\
                plot_roc=False, plot_conf_matrix=False, show_summary=False,\
                ret_eval_dict=True)

    test_pred_dir_ds = test_ds.copy(deepcopy=True)
    test_pred_dir_ds.labels = cls_mdls['lgb_1_dir']['preds_test'].reshape(-1,1)
    metrics_test_dir_dict, _ =\
            mldatasets.compute_aif_metrics(test_ds, test_pred_dir_ds,\
                                unprivileged_groups=unprivileged_groups,\
```

Prejudice Remover Method

```
[ ] log_pr_mdl = PrejudiceRemover(eta=1.0, sensitive_attr='AGE_GROUP', class_attr='IS_DEFAULT')
    log_pr_mdl.fit(train_ds)
```

```
<aif360.algorithms.inprocessing.prejudice_remover.PrejudiceRemover at 0x7fde28021d00>
```

```
[ ] train_pred_pr_ds = log_pr_mdl.predict(train_ds)
    test_pred_pr_ds = log_pr_mdl.predict(test_ds)

    cls_mdls['log_2_pr'] =\
        mldatasets.evaluate_class_metrics_mdl(log_pr_mdl, train_pred_pr_ds.labels,\
                    test_pred_pr_ds.scores, test_pred_pr_ds.labels, y_train, y_test)
    metrics_test_pr_dict, _ =\
            mldatasets.compute_aif_metrics(test_ds, test_pred_pr_ds,\
                                unprivileged_groups=unprivileged_groups,\
                                privileged_groups=privileged_groups)
    cls_mdls['log_2_pr'].update(metrics_test_pr_dict)
```

GerryFair Classifier Method

```
[ ] dt_gf_mdl = GerryFairClassifier(C=100, gamma=.005, fairness_def='FN',\
                                    max_iters=50, printflag=True,\
                                    predictor=tree.DecisionTreeRegressor(max_depth=3))

    dt_gf_mdl.fit(train_ds, early_termination=True)
```

```
iteration: 1, error: 0.17863181465064204, fairness violation: 0.00248572931140393, violated group size: 0.1340309829548051
iteration: 2, error: 0.17863181465064204, fairness violation: 0.00248572931140393, violated group size: 0.1340309829548051
iteration: 3, error: 0.17863181465064204, fairness violation: 0.00248572931140393, violated group size: 0.1340309829548051
iteration: 4, error: 0.17863181465064204, fairness violation: 0.00248572931140393, violated group size: 0.1340309829548051
iteration: 5, error: 0.17863181465064204, fairness violation: 0.00248572931140393, violated group size: 0.1340309829548051
<aif360.algorithms.inprocessing.gerryfair_classifier.GerryFairClassifier at 0x7fde6065fd30>
```

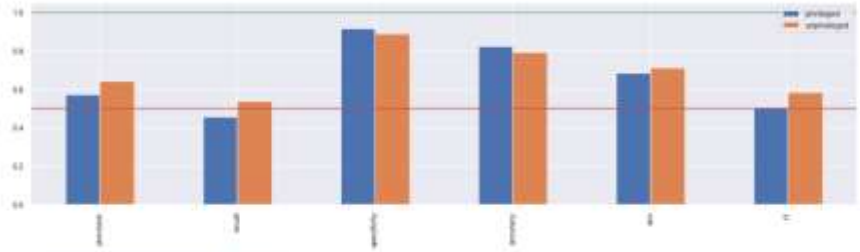
```
[ ] train_pred_gf_ds = dt_gf_mdl.predict(train_ds, threshold=False)
    test_pred_gf_ds = dt_gf_mdl.predict(test_ds, threshold=False)
    cls_mdls['dt_2_gf'] =\
```

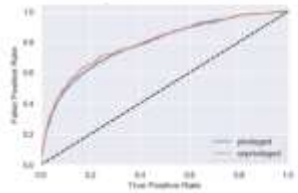
Tying It All Together!

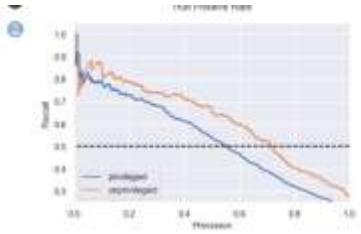
```
cls_metrics_df = pd.DataFrame.from_dict(cls_mdls, 'index')\
                    [['accuracy_train', 'accuracy_test', 'f1_test', 'mcc_test',\
                      'SPD', 'DI', 'AOD', 'EOD', 'DFBA']]
with pd.option_context('display.precision', 4):
    html = cls_metrics_df.sort_values(by='accuracy_test', ascending=False).style.\
        background_gradient(cmap='plasma_r', low=0.3, high=1, subset=['SPD', 'AOD', 'EOD']).\
        background_gradient(cmap='viridis_r', low=1, high=0.3, subset=['DI', 'DFBA'])
html
```

	accuracy_train	accuracy_test	f1_test	mcc_test	SPD	DI	AOD	EOD	DFBA
dt_2_gf	0.8214	0.8262	0.4812	0.4136	-0.0548	0.8368	-0.0430	-0.0219	0.2521
lgb_0_base	0.8255	0.8167	0.5330	0.4243	-0.0679	0.8193	-0.0550	-0.0265	0.2328
lgb_1_rw	0.8240	0.8138	0.5337	0.4210	-0.0371	0.9552	-0.0171	-0.0018	0.0348
lgb_1_dir	0.8237	0.8129	0.5301	0.4171	-0.0824	0.8252	-0.0483	-0.0214	0.2545
lgb_3_epp	0.8255	0.8101	0.5152	0.4025	-0.0260	0.9668	0.0022	-0.0021	0.0031
lgb_3_cpp	0.8255	0.2622	0.2129	-0.3055	-0.0711	0.7609	-0.0635	-0.1262	0.0432
log_2_pr	0.1912	0.1873	0.2844	-0.3383	0.0520	1.7627	0.0498	0.0295	0.3454

```
test_df = ccdefault_bias_df.loc[X_test.index]
test_df['AGE_GROUP'] =\
    test_df.AGE_GROUP.replace({0:'unprivileged', 1:'privileged'})
cat_cols_1 = ccdefault_bias_df.dtypes[lambda x: x==np.int8].index.tolist()

_ = xai.metrics_plot(
        y_test, cls_mdls['lgb_3_epp']['probs_test'],\
        df=test_df, cross_cols=['AGE_GROUP'],\
        categorical_cols=cat_cols_1)
_ = xai.roc_plot(
        y_test, cls_mdls['lgb_3_epp']['probs_test'],\
        df=test_df, cross_cols=['AGE_GROUP'],\
        categorical_cols=cat_cols_1)
_ = xai.pr_plot(
        y_test, cls_mdls['lgb_3_epp']['probs_test'],\
        df=test_df, cross_cols=['AGE_GROUP'],\
        categorical_cols=cat_cols_1)
```







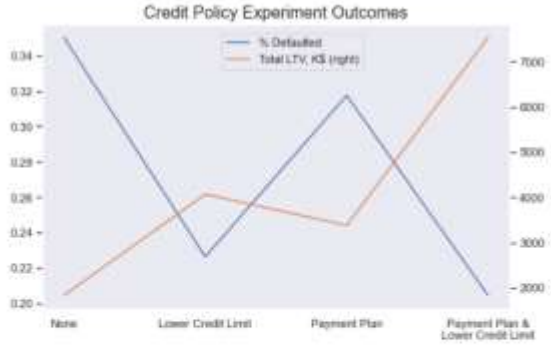
Creating a Causal Model

```
treatment_names = ['Lower Credit Limit', 'Payment Plan', 'Payment Plan &Lower Credit Limit']
all_treatment_names = np.array(["None"] + treatment_names)
```

Understanding the Results of the Experiment

```
pct_x = ccdefault_causal_df[ccdefault_causal_df.IS_DEFAULT==1].\
                groupby(['_TREATMENT']).size() /\
            ccdefault_causal_df.groupby(['_TREATMENT']).size()
ltv_x = ccdefault_causal_df.groupby(['_TREATMENT'])\
                                        ['_LTV'].sum()/1000
plot_df = pd.DataFrame({'% Defaulted':pct_x, 'Total LTV, K$':ltv_x})
plot_df.index = all_treatment_names

ax = plot_df.plot(secondary_y=['Total LTV, K$'], figsize=(8,5))
ax.get_legend().set_bbox_to_anchor((0.7, 0.99))
plt.grid(False)
plt.title("Credit Policy Experiment Outcomes", fontsize=16)
plt.show()
```

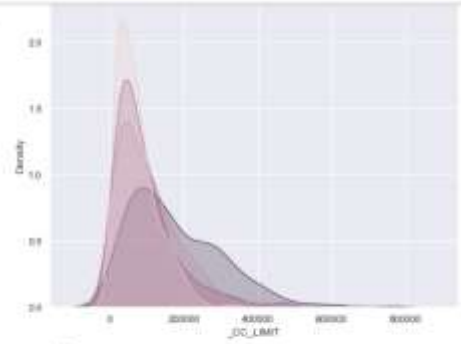
[ ]

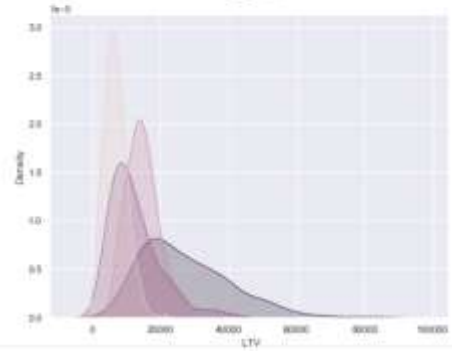


[ ]
```
plt.figure(figsize=(10, 7))
sns.scatterplot(
    x=ccdefault_causal_df['_CC_LIMIT'].values,
    y=ccdefault_causal_df['_risk_score'].values,
    hue=all_treatment_names[ccdefault_causal_df['_TREATMENT'].values],
    hue_order=all_treatment_names
)
plt.title("Chosen Credit Policy ('Treatment') by Customer", fontsize=16)
plt.xlabel("Original Credit Limit", fontsize=14)
plt.ylabel("Risk Factor", fontsize=14)
plt.show()
```

[ ]
```
ax = sns.displot(ccdefault_causal_df, x="_CC_LIMIT", hue="_TREATMENT", kind="kde", fill=True)
ax.fig.set_figheight(6)
ax.fig.set_figwidth(10)
ax = sns.displot(ccdefault_causal_df, x="_LTV", hue="_TREATMENT", kind="kde", fill=True)
ax.fig.set_figheight(6)
ax.fig.set_figwidth(10)
```

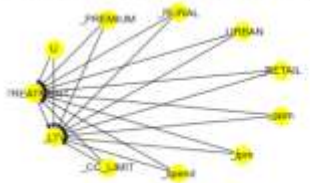
[ ]



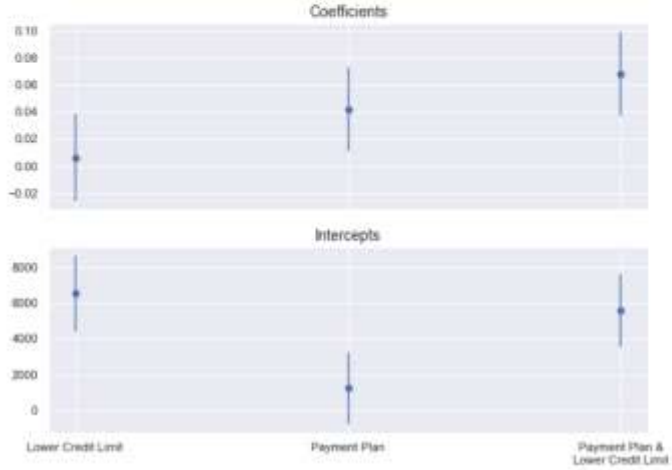


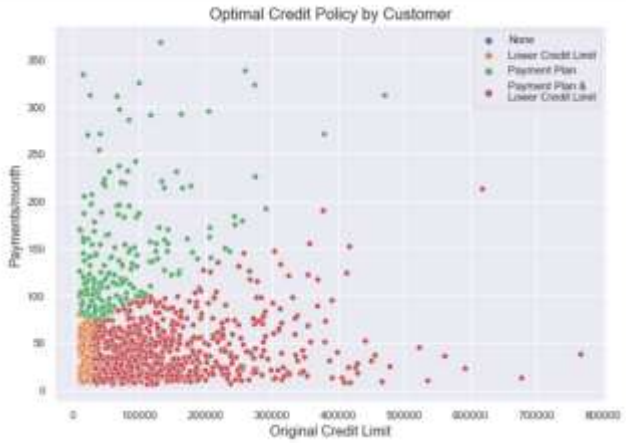

```
[ ] try:
        display(
            Image(to_pydot(causal_mdl._graph._graph).create_png())
        )
        #to_pydot(causal_mdl._graph._graph).write('causal_model.dot')
    except:
        causal_mdl.view_model()
```

```
WARNING:dowhy.causal_graph:Warning: Pygraphviz cannot be loaded. Check that graphviz and pygraphviz are installed.
INFO:dowhy.causal_graph:Using Matplotlib for plotting
```

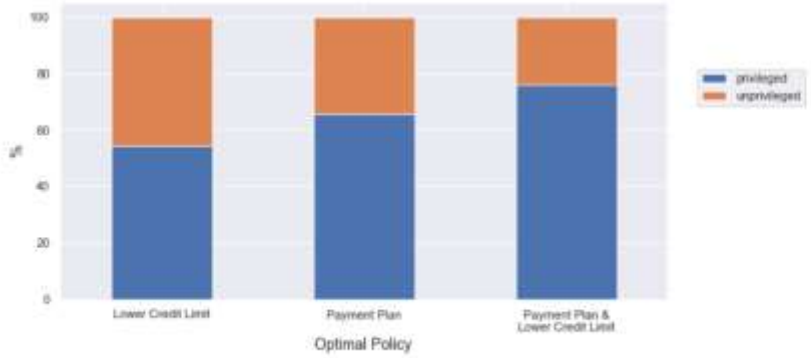


```
[ ] plt.figure(figsize=(10, 7))
    ax1 = plt.subplot(2, 1, 1)
    plt.errorbar(idxs, coefs, coefs_err, fmt="o")
    plt.xticks(idxs, treatment_names)
    plt.setp(ax1.get_xticklabels(), visible=False)
    plt.title("Coefficients", fontsize=14)

    plt.subplot(2, 1, 2)
    plt.errorbar(idxs, intercepts, intercepts_err, fmt="o")
    plt.xticks(idxs, treatment_names)
    plt.title("Intercepts", fontsize=14)
    plt.show()
```





```
[ ]  ccdefault_causal_df['recommended_T'] = recommended_T
     plot_df = ccdefault_causal_df.groupby(['recommended_T','AGE_GROUP']).\
                                                    size().reset_index()
     plot_df['AGE_GROUP'] = plot_df.AGE_GROUP.replace({0:'unprivileged', 1:'privileged'})
     plot_df = plot_df.pivot(columns='AGE_GROUP', index='recommended_T', values=0)
     plot_df.index = treatment_names
     plot_df = plot_df.apply(lambda r: r/r.sum()*100, axis=1)

     plot_df.plot.bar(stacked=True, rot=0, figsize=(10,5))
     plt.legend(loc=(1.05, 0.65))
     plt.xlabel('Optimal Policy', fontsize=14)
     plt.ylabel('%', fontsize=14)
     plt.show()
```



Chapter 12 (Recidivism):

```
!!pip install --upgrade pandas numpy scikit-learn xgboost lightgbm tensorflow pydot graphviz matplotlib seaborn scipy
     ---------------------------------------- 865.6/865.6 KB 57.2 MB/s eta 0:00:00
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (1.4.4)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (7.1.2)
Collecting contourpy>=1.0.1
  Downloading contourpy-1.0.7-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (300 kB)
     ---------------------------------------- 300.9/300.9 kB 20.8 MB/s eta 0:00:00
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.8/dist-packages (from tensorboard<2.12,>=2.11->tensorflow) (3.4.1)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.8/dist-packages (from tensorboard<2.12,>=2.11->tensorflow) (2.25.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.8/dist-packages (from tensorboard<2.12,>=2.11->tensorflow) (1.34.0)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from tensorboard<2.12,>=2.11->tensorflow) (1.0.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.8/dist-packages (from tensorboard<2.12,>=2.11->tensorflow) (1.8.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.8/dist-packages (from tensorboard<2.12,>=2.11->tensorflow) (0.6.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.8/dist-packages (from tensorboard<2.12,>=2.11->tensorflow) (0.4.6)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.8/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.8/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow) (4.9)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow) (5.3.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.8/dist-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tensorflow) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.8/dist-packages (from markdown>=2.6.8->tensorboard<2.12,>=2.11->tensorflow) (6.0.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (2022.12.7)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (4.0.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.8/dist-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.12,>=2.11->tensorflow) (3.11.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.8/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.8/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tensorflow) (3.2.2)
Installing collected packages: flatbuffers, tensorflow-estimator, pydot, numpy, keras, graphviz, fonttools, scipy, pandas, contourpy, xgboost, scikit-learn, matplotlib, tensorboard, seaborn, lightgbm, tensorflow
  Attempting uninstall: flatbuffers
    Found existing installation: flatbuffers 1.12
    Uninstalling flatbuffers-1.12:
      Successfully uninstalled flatbuffers-1.12
  Attempting uninstall: tensorflow-estimator
    Found existing installation: tensorflow-estimator 2.5.0
    Uninstalling tensorflow-estimator-2.5.0:
      Successfully uninstalled tensorflow-estimator-2.5.0
  Attempting uninstall: pydot
    Found existing installation: pydot 1.1.0
    Uninstalling pydot-1.1.0:
      Successfully uninstalled pydot-1.1.0
  Attempting uninstall: numpy
    Found existing installation: numpy 1.21.6
    Uninstalling numpy-1.21.6:
```

```
!pip install --upgrade machine-learning-datasets
!pip install --upgrade catboost bayesian-optimization tensorflow-lattice shap
!pip install --no-deps git+https://github.com/EthicalML/xai.git
```

ding the Libraries

```
import math
import os
import copy
import machine_learning_datasets as mldatasets
import pandas as pd
import numpy as np
from sklearn import preprocessing, model_selection, metrics,\
                    linear_model, svm, neural_network, ensemble
import xgboost as xgb
import lightgbm as lgb
import catboost as cb
import tensorflow as tf
from bayes_opt import BayesianOptimization
import tensorflow_lattice as tfl
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
import xai
import shap
```

```
print(tf.__version__)
```

```
2.2.1
```

Understanding and Preparing the Data

```
[ ] recidivism_df = mldatasets.load("recidivism-risk-balanced")
```

```
https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python/raw/master/datasets/recidivism_risk_balanced.csv downloaded to /Users/rawals/Documents/Personal/Interpretable Machine Learning with Python/Chapter12/data folder
1 dataset files found in /Users/rawals/Documents/Personal/Interpretable Machine Learning with Python/Chapter12/data folder
parsing /Users/rawals/Documents/Personal/Interpretable Machine Learning with Python/Chapter12/data/recidivism_risk_balanced.csv
```

```
[ ] recidivism_df.info()
```

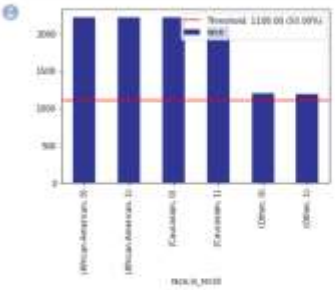
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11762 entries, 0 to 11761
Data columns (total 12 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   sex                     11762 non-null  object
 1   age                     11762 non-null  int8
 2   race                    11762 non-null  object
 3   juv_fel_count           11762 non-null  int32
 4   juv_misd_count          11762 non-null  int32
 5   juv_other_count         11762 non-null  int32
 6   priors_count            11762 non-null  int32
 7   c_charge_degree         11762 non-null  object
 8   days_b_screening_arrest 11762 non-null  float64
 9   length_of_stay          11762 non-null  float64
 10  compas_score            11762 non-null  int32
 11  is_recid                11762 non-null  int32
dtypes: float64(2), int32(7), object(3)
memory usage: 911.5+ KB
```

Verifying the Sampling Balance

```
categorical_cols_1 = ['sex', 'race', 'c_charge_degree',\
                      'is_recid', 'compas_score']
xai.imbalance_plot(recidivism_df, 'race', 'is_recid',\
                   categorical_cols=categorical_cols_1)
```

Verifying the Sampling Balance

```
categorical_cols_1 = ['sex', 'race', 'c_charge_degree',\
                      'is_recid', 'compas_score']
xai.imbalance_plot(recidivism_df, 'race', 'is_recid',\
                   categorical_cols=categorical_cols_1)
```




```
[ ] recidivism_corr_df = recidivism_df.drop(['compas_score'], axis=1)
    pd.DataFrame({'feature': recidivism_corr_df.columns[:-1],\
                  'correlation_to_target': scipy.stats.spearmanr(recidivism_corr_df).correlation[10,:-1]\
                 }).style.background_gradient(cmap='coolwarm')
```

	feature	correlation_to_target
0	sex	0.093255
1	age	-0.166938
2	race	-0.004998
3	juv_fel_count	0.082138
4	juv_misd_count	0.117576
5	juv_other_count	0.106293

Placing Guardrails with Feature Engineering

Ordinalization

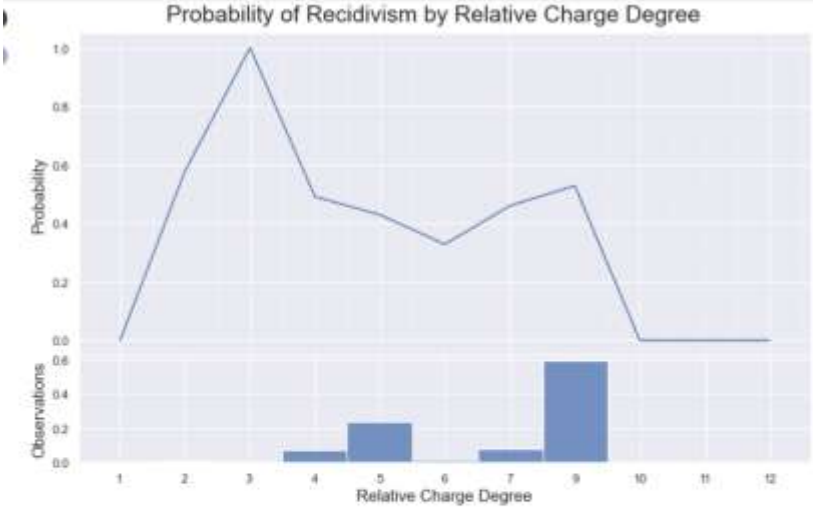
```
[ ] recidivism_df.c_charge_degree.value_counts()
```

```
(F3)     6555
(M1)     2632
(F2)      857
(M2)      760
(F1)      131
(F7)      104
(MO3)      76
(F5)        7
(F6)        5
(NI0)       4
(CO3)       2
(TCX)       1
Name: c_charge_degree, dtype: int64
```

```
[ ] charge_degree_code_rank = {'(F10)': 15, '(F9)':14, '(F8)':13, '(F7)':12, '(TCX)':11, '(F6)':10, '(F5)':9,\
                               '(F4)':8, '(F3)':7, '(F2)':6, '(F1)':5, '(M1)':4, '(NI0)':4, '(M2)':3,\
                               '(CO3)':2, '(MO3)':1, '(X)':0}
    recidivism_df.c_charge_degree.replace(charge_degree_code_rank, inplace=True)
```

```
[ ] mldatasets.plot_prob_progression(recidivism_df.c_charge_degree, recidivism_df.is_recid,\
                        x_intervals=12, use_quartiles=False, xlabel='Relative Charge Degree',\
                        title='Probability of Recidivism by Relative Charge Degree')
```

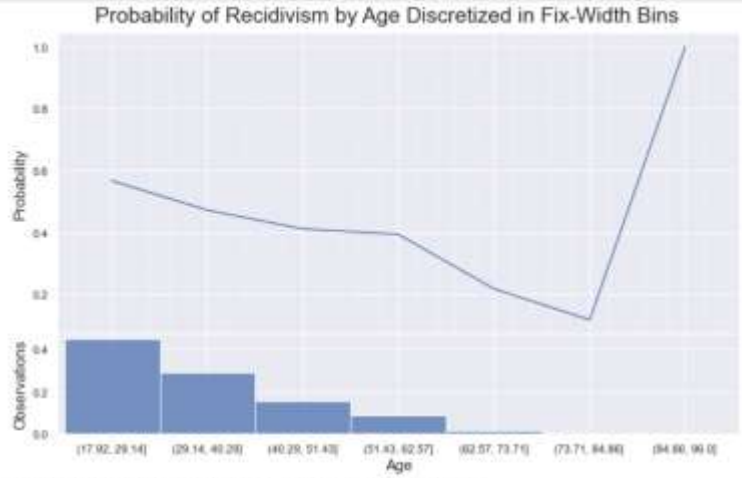
```
elementwise comparison failed; returning scalar instead, but in the future will perform elementwise comparison
FixedFormatter should only be used together with FixedLocator
```



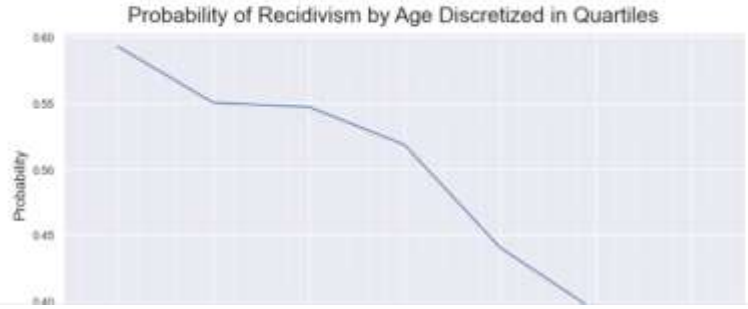
scretization

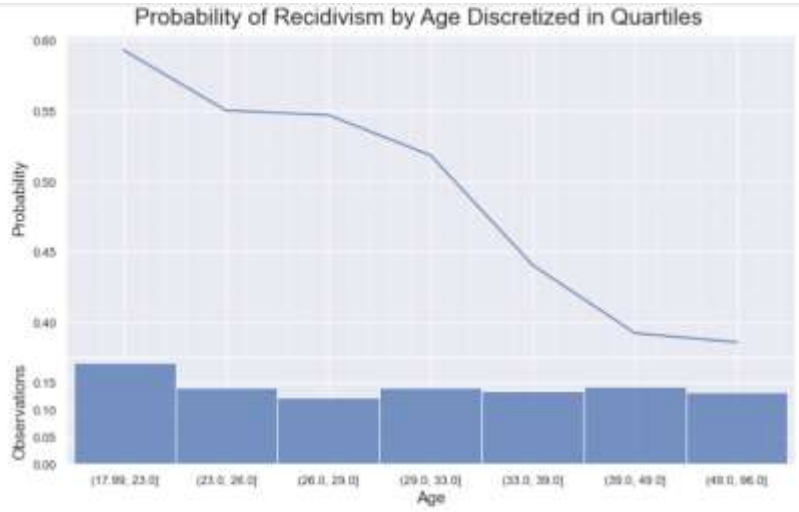
```
] mldatasets.plot_prob_progression(recidivism_df.age, recidivism_df.is_recid,\
                                   x_intervals=7, use_quartiles=False, xlabel='Age',\
                    title='Probability of Recidivism by Age Discretized in Fix-Width Bins')

  mldatasets.plot_prob_progression(recidivism_df.age, recidivism_df.is_recid,\
                                   x_intervals=7, use_quartiles=True, xlabel='Age',\
                    title='Probability of Recidivism by Age Discretized in Quartiles')
```

```
FixedFormatter should only be used together with FixedLocator
```




```
FixedFormatter should only be used together with FixedLocator
```



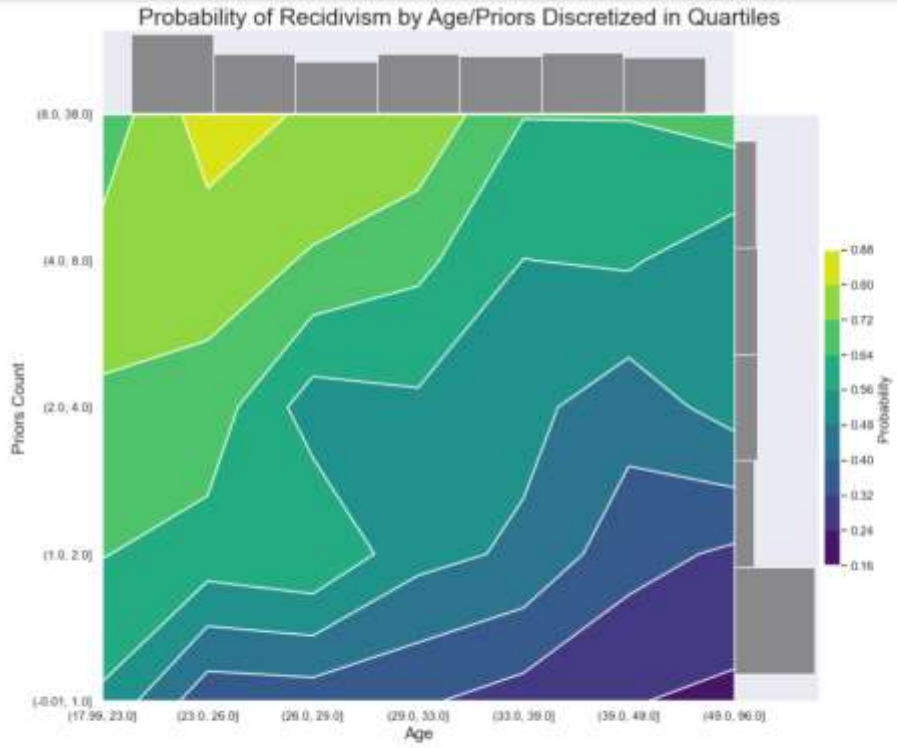



```
recidivism_df['age_group'] = pd.qcut(recidivism_df.age, 7, precision=0).astype(str)
```

eraction Terms and Non-linear Transformations

```
mldatasets.plot_prob_contour_map(recidivism_df.age, recidivism_df.priors_count, recidivism_df.is_recid,\
                                 use_quartiles=True, xlabel='Age', ylabel='Priors Count',\
                    title='Probability of Recidivism by Age/Priors Discretized in Quartiles')
```

```
FixedFormatter should only be used together with FixedLocator
FixedFormatter should only be used together with FixedLocator
```

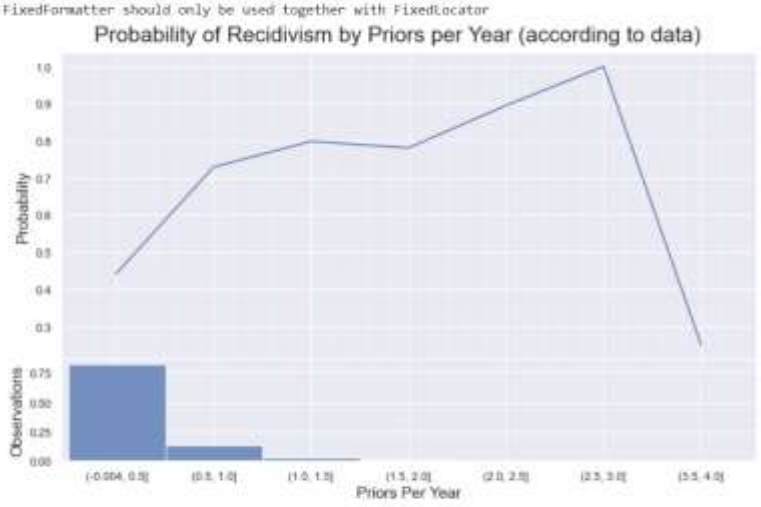



```
recidivism_df['priors_per_year'] = recidivism_df['priors_count']/(recidivism_df['age'] - 17)
```

```
mldatasets.plot_prob_progression(recidivism_df.priors_per_year, recidivism_df.is_recid,\
```

```
mldatasets.plot_prob_progression(recidivism_df.priors_per_year, recidivism_df.is_recid,\
                                 x_intervals=8, xlabel='Priors Per Year',\
            title='Probability of Recidivism by Priors per Year (according to data)')
```

```
FixedFormatter should only be used together with FixedLocator
```

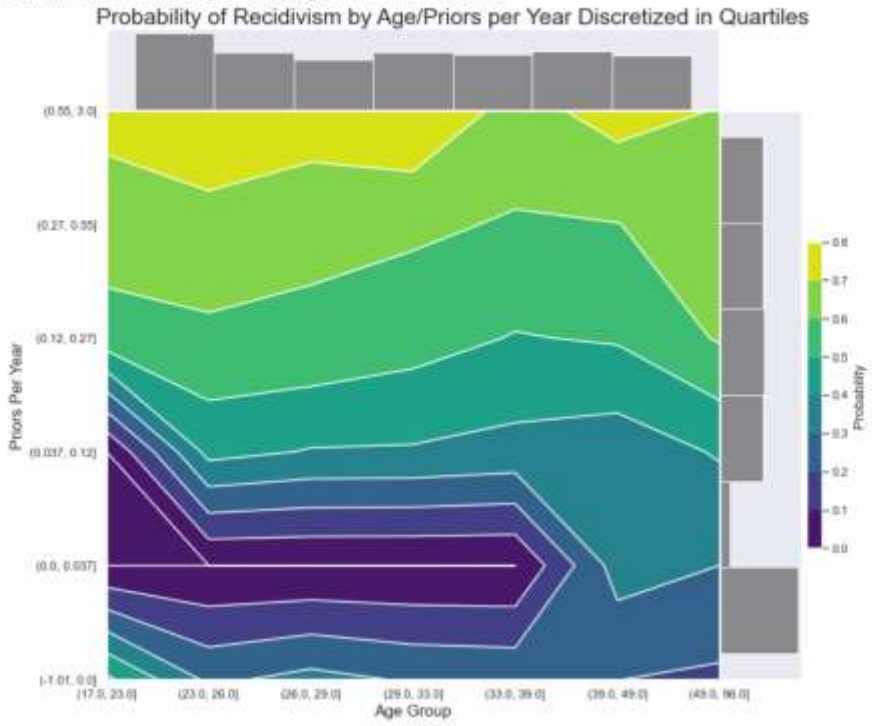



```
recidivism_df.loc[recidivism_df.priors_per_year > 3, 'priors_per_year'] = -1
```

```
mldatasets.plot_prob_contour_map(recidivism_df.age_group, recidivism_df.priors_per_year,\
                                 recidivism_df.is_recid, y_intervals=6, use_quartiles=True,\
                                 xlabel='Age Group', ylabel='Priors Per Year',\
            title='Probability of Recidivism by Age/Priors per Year Discretized in Quartiles')
```

```
elementwise comparison failed; returning scalar instead, but in the future will perform elementwise comparison
FixedFormatter should only be used together with FixedLocator
```

```
elementwise comparison failed; returning scalar instead, but in the future will perform elementwise comparison
FixedFormatter should only be used together with FixedLocator
FixedFormatter should only be used together with FixedLocator
```



Categorical Encoding

```
[ ] cat_feat_l = ['sex', 'race', 'age_group']
    ordenc = preprocessing.OrdinalEncoder(dtype=np.int8)
    recidivism_df[cat_feat_l] = ordenc.fit_transform(recidivism_df[cat_feat_l])
    recidivism_df.drop(['age', 'priors_count', 'compas_score'], axis=1, inplace=True)
```

Other Preparations

```
[ ] rand = 9
    os.environ['PYTHONHASHSEED'] = str(rand)
    tf.random.set_seed(rand)
    np.random.seed(rand)

    y = recidivism_df['is_recid']
    X = recidivism_df.drop(['is_recid'], axis=1).copy()
    X_train, X_test, y_train, y_test =\
            model_selection.train_test_split(X, y, test_size=0.2, random_state=rand)
    recidivism_df = X.join(y)
```

```
[ ] pd.DataFrame({'feature': X.columns,\
                  'correlation_to_target': scipy.stats.spearmanr(recidivism_df).correlation[10,:-1]\
                 }).style.background_gradient(cmap='coolwarm')
```

	feature	correlation_to_target
0	sex	0.093265
1	race	-0.004598
2	juv_fel_count	0.082138
3	juv_misd_count	0.117976
4	juv_other_count	0.125797
5	c_charge_degree	0.068603
6	days_b_screening_arrest	0.032485
7	length_of_stay	0.012530
8	age_group	-0.152131
9	priors_per_year	0.321885

Define the Model and Parameters to Tune

```
[ ] def build_nn_mdl(hidden_layer_sizes, l1_reg=0, l2_reg=0, dropout=0):
        nn_model = tf.keras.Sequential([
            tf.keras.Input(shape=[len(X_train.keys())]),
            tf.keras.layers.experimental.preprocessing.Normalization()
        ])
        reg_args = {}
        if (l1_reg > 0) or (l2_reg > 0):
            reg_args = {'kernel_regularizer':\
                        tf.keras.regularizers.l1_l2(l1=l1_reg, l2=l2_reg)}
        for hidden_layer_size in hidden_layer_sizes:
            nn_model.add(tf.keras.layers.Dense(hidden_layer_size,\
                                               activation='relu', **reg_args))
        if dropout > 0:
            nn_model.add(tf.keras.layers.Dropout(dropout))
        nn_model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

        nn_model.compile(loss='binary_crossentropy',\
                    optimizer=tf.keras.optimizers.Adam(lr=0.0004),\
                    metrics=['accuracy',tf.keras.metrics.AUC(name='auc')])

        return nn_model
```

Run the Hyperparameter Tuning

```
[ ] cv = model_selection.RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=rand)
    nn_grid = {'hidden_layer_sizes':[(80,)], 'l1_reg':[0,0.005],\
               'l2_reg':[0,0.01], 'dropout':[0,0.05]}
    nn_model = KerasClassifier(build_fn=build_nn_mdl)
    nn_grid_search = model_selection.GridSearchCV(estimator=nn_model, cv=cv, n_jobs=-1,\
                                                  param_grid=nn_grid, scoring='precision',\
                                                  error_score=0, verbose=0)
    nn_grid_result = nn_grid_search.fit(X_train.astype(float), y_train.astype(float),\
                                        epochs=400, batch_size=128, verbose=0)
```

Examine the Results

```
[ ] print(nn_grid_result.best_params_)
```

```
{'dropout': 0.05, 'hidden_layer_sizes': (80,), 'l1_reg': 0.005, 'l2_reg': 0.01}
```

```
[ ] pd.DataFrame(nn_grid_result.cv_results_)\
        [['param_hidden_layer_sizes','param_l1_reg', 'param_l2_reg', 'param_dropout',\
          'mean_test_score', 'std_test_score', 'rank_test_score']].\
        sort_values(by='rank_test_score')
```

	param_hidden_layer_sizes	param_l1_reg	param_l2_reg	param_dropout	mean_test_score	std_test_score	rank_test_score
7	(80,)	0.005	0.01	0.05	0.876763	0.020052	1
6	(80,)	0.005	0	0.05	0.868818	0.016849	2
3	(80,)	0.005	0.01	0	0.868501	0.022394	3
5	(80,)	0	0.01	0.05	0.867027	0.016773	4
1	(80,)	0	0.01	0	0.865925	0.018512	5
2	(80,)	0.005	0	0	0.865502	0.023206	6
4	(80,)	0	0	0.05	0.864336	0.013002	7
0	(80,)	0	0	0	0.861700	0.015193	8

Evaluate the Best Model

```
[ ] sns.set()
    fitted_class_mdls = {}
    fitted_class_mdls['keras_reg'] =\
           mldatasets.evaluate_class_mdl(nn_grid_result.best_estimator_, X_train.astype(float),\
                     X_test.astype(float), y_train.astype(float), y_test.astype(float),\
                     plot_roc=False, plot_conf_matrix=True, ret_eval_dict=True)
    fitted_class_mdls['keras_reg']['cv_results'] =\
                                    nn_grid_result.cv_results_
    fitted_class_mdls['keras_reg']['cv_best_params'] =\
```

Tuning Other Popular Model Classes

Batch Hyperparameter Tuning Models

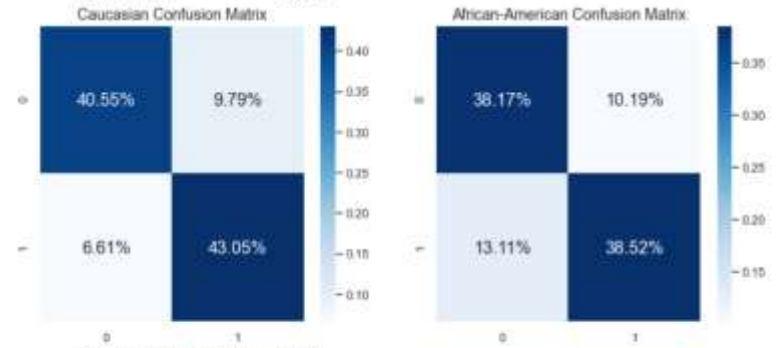
```
[ ]  class_mdls = {
        'logistic':{'model':linear_model.LogisticRegression(random_state=rand,\
                                                             max_iter=1000),\
                    'grid':{'C':np.linspace(0.01, 0.49, 25),\
                            'class_weight':[{0:6,1:5}],\
                            'solver':['lbfgs', 'liblinear', 'newton-cg']}},\
        'svc':{'model':svm.SVC(probability=True, random_state=rand),\
               'grid':{'C':[15,25,40], 'class_weight':[{0:6,1:5}]}},\
        'nu-svc':{'model':svm.NuSVC(probability=True, random_state=rand),\
                  'grid':{'nu':[0.2,0.3], 'gamma':[0.6,0.7],\
                          'class_weight':[{0:6,1:5}]}},\
        'mlp':{'model':neural_network.MLPClassifier(random_state=rand,\
                            hidden_layer_sizes=(80,), early_stopping=True),\
               'grid':{'alpha':np.linspace(0.05, 0.15, 11),\
                       'activation':['relu','tanh','logistic']}},\
        'rf':{'model':ensemble.RandomForestClassifier(random_state=rand,\
                            max_depth=7, oob_score=True, bootstrap=True),\
              'grid':{'max_features':[6,7,8], 'max_samples':[0.75,0.9,1],\
                      'class_weight':[{0:6,1:5}]}},\
        'xgb-rf':{'model':xgb.XGBRFClassifier(seed=rand, eta=1, max_depth=7,
                                              n_estimators=200),\
                  'grid':{'scale_pos_weight':[0.85], 'reg_lambda':[1,1.5,2],\
                          'reg_alpha':[0,0.5,0.75,1]}},\
        'xgb':{'model':xgb.XGBClassifier(seed=rand, eta=1, max_depth=7),\
               'grid':{'scale_pos_weight':[0.7], 'reg_lambda':[1,1.5,2],\
                       'reg_alpha':[0.5,0.75,1]}},\
        'lgbm':{'model':lgb.LGBMClassifier(random_seed=rand, learning_rate=0.7,\
                                           max_depth=5),\
                'grid':{'lambda_l2':[0,0.5,1], 'lambda_l1':[0,0.5,1],\
                        'scale_pos_weight':[0.8]}},\
        'catboost':{'model':cb.CatBoostClassifier(random_seed=rand, depth=5,\
                                                  learning_rate=0.5, verbose=0),\
                    'grid':{'l2_leaf_reg':[2,2.5,3], 'scale_pos_weight':[0.65]}}
    }
```

Evaluate Models by Precision

```
[ ] class_metrics = pd.DataFrame.from_dict(fitted_class_mdls,\
                    'index')[['accuracy_train', 'accuracy_test',\
                              'precision_train', 'precision_test',\
                              'recall_train', 'recall_test',\
                              'roc-auc_test', 'f1_test', 'mcc_test']]
    with pd.option_context('display.precision', 3):
        html = class_metrics.sort_values(by='precision_test', ascending=False).\
            style.background_gradient(cmap='plasma', subset=['precision_test']).\
            background_gradient(cmap='viridis', subset=['recall_test'])
    html
```

	accuracy_train	accuracy_test	precision_train	precision_test	recall_train	recall_test	roc-auc_test	f1_test	mcc_test
catboost_reg	0.964	0.820	0.992	0.837	0.935	0.802	0.881	0.819	0.641
nu-svc_reg	0.939	0.807	0.950	0.836	0.925	0.772	0.858	0.803	0.616
xgb_reg	0.966	0.820	0.988	0.829	0.943	0.815	0.877	0.821	0.639
xgb_base	0.978	0.823	0.984	0.812	0.967	0.848	0.880	0.830	0.646
catboost_base	0.970	0.817	0.980	0.809	0.958	0.838	0.879	0.823	0.634
lgbm_reg	0.871	0.761	0.913	0.798	0.815	0.709	0.831	0.751	0.526
lgbm_base	0.871	0.756	0.881	0.786	0.852	0.748	0.825	0.767	0.512
logistic_reg	0.643	0.638	0.721	0.745	0.444	0.437	0.701	0.551	0.309
svc_reg	0.623	0.648	0.741	0.728	0.356	0.401	0.716	0.586	0.317
mlp_reg	0.649	0.653	0.690	0.724	0.518	0.514	0.706	0.601	0.324
rf_reg	0.722	0.686	0.741	0.716	0.668	0.636	0.759	0.673	0.376
logistic_base	0.651	0.662	0.686	0.712	0.535	0.531	0.701	0.608	0.318
keras_reg	0.660	0.648	0.683	0.707	0.534	0.525	0.694	0.603	0.310
xgb-rf_reg	0.719	0.681	0.730	0.703	0.678	0.643	0.748	0.672	0.363
xgb-rf_base	0.722	0.699	0.706	0.696	0.742	0.718	0.760	0.706	0.397
mlp_base	0.658	0.658	0.688	0.689	0.604	0.596	0.714	0.639	0.320
rf_base	0.708	0.681	0.698	0.687	0.714	0.686	0.753	0.686	0.363
svc_base	0.584	0.568	0.579	0.583	0.560	0.523	0.608	0.552	0.137
nu-svc_base	0.531	0.508	0.560	0.580	0.204	0.122	0.579	0.201	0.049


```
African-American FPR:   16.9%
Caucasian FPR:          15.6%
Ratio FPRs:             1.09 x
```




```
African-American FPR:   21.1%
Caucasian FPR:          19.5%
Ratio FPRs:             1.08 x
```

Running Bayesian Hyperparameter Tuning

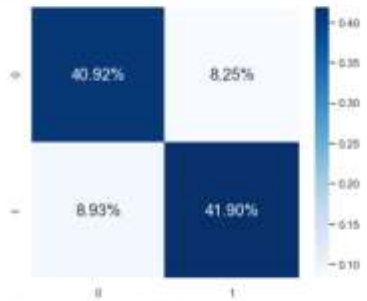
```
[ ] def hyp_catboost(l2_leaf_reg, scale_pos_weight):
        cv = model_selection.RepeatedStratifiedKFold(n_splits=4,\
                                    n_repeats=3, random_state=rand)
        metric_l = []
        for train_index, val_index in cv.split(X_train, y_train):
            X_train_cv, X_val_cv = X_train.iloc[train_index], X_train.iloc[val_index]
            y_train_cv, y_val_cv = y_train.iloc[train_index], y_train.iloc[val_index]
            mdl = cb.CatBoostClassifier(random_seed=rand,learning_rate=0.5, verbose=0,\
                                        depth=5, l2_leaf_reg=l2_leaf_reg,\
                                        scale_pos_weight=scale_pos_weight)
            mdl = mdl.fit(X_train_cv, y_train_cv)
            y_val_pred = mdl.predict(X_val_cv)
            metric = weighted_penalized_pr_average(y_val_cv, y_val_pred,\
                                                   X_val_cv['race'], range(3))
            metric_l.append(metric)
        return np.median(np.array(metric_l))
```

```
[ ] pbounds = {
                'l2_leaf_reg': (2,4),
                'scale_pos_weight': (0.55,0.85)
              }
    optimizer = BayesianOptimization(hyp_catboost, pbounds, random_state=rand)
    optimizer.maximize(init_points=3, n_iter=7)
```

```
|   iter    |  target   | l2_lea... | scale_... |
-------------------------------------------------
|  1        |  0.7683   |  2.021    |  0.7006   |
|  2        |  0.7656   |  2.992    |  0.5901   |
|  3        |  0.766    |  2.284    |  0.6356   |
|  4        |  0.7632   |  2.744    |  0.8045   |
|  5        |  0.7594   |  2.917    |  0.6475   |
|  6        |  0.7653   |  3.787    |  0.7465   |
|  7        |  0.7637   |  3.043    |  0.8094   |
|  8        |  0.761    |  2.162    |  0.5572   |
|  9        |  0.7567   |  2.047    |  0.6749   |
|  10       |  0.7606   |  3.583    |  0.6271   |
=================================================
```

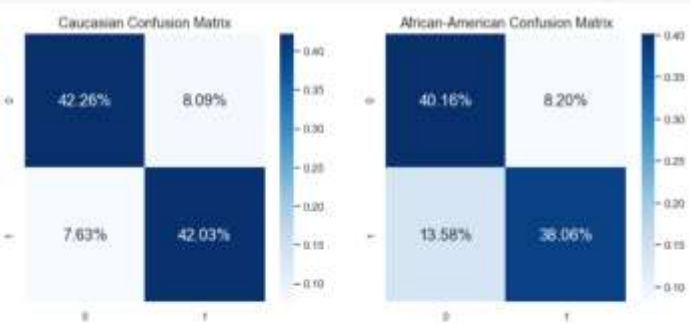
Fitting and Evaluating a Model with the Best Parameters

```
[ ] cb_opt = cb.CatBoostClassifier(random_seed=rand, depth=5,learning_rate=0.5,\
                                   verbose=0, **optimizer.max['params'])
    cb_opt = cb_opt.fit(X_train, y_train)
    fitted_class_mdls['catboost_opt'] =\
            mldatasets.evaluate_class_mdl(cb_opt, X_train, X_test, y_train, y_test,\
                      plot_roc=False, plot_conf_matrix=True, ret_eval_dict=True)
```




```
Accuracy_train:  0.9721          Accuracy_test:   0.8282
Precision_test:  0.8354          Recall_test:     0.8244
ROC-AUC_test:    0.8815          F1_test:         0.8299          MCC_test: 0.6564
```

```
[ ] y_test_pred = fitted_class_mdls['catboost_opt']['preds_test']
    _ = mldatasets.\
            compare_confusion_matrices(y_test[X_test.race==1], y_test_pred[X_test.race==1],\
                                       y_test[X_test.race==0], y_test_pred[X_test.race==0],\
                                       'Caucasian', 'African-American', compare_fpr=True)
```



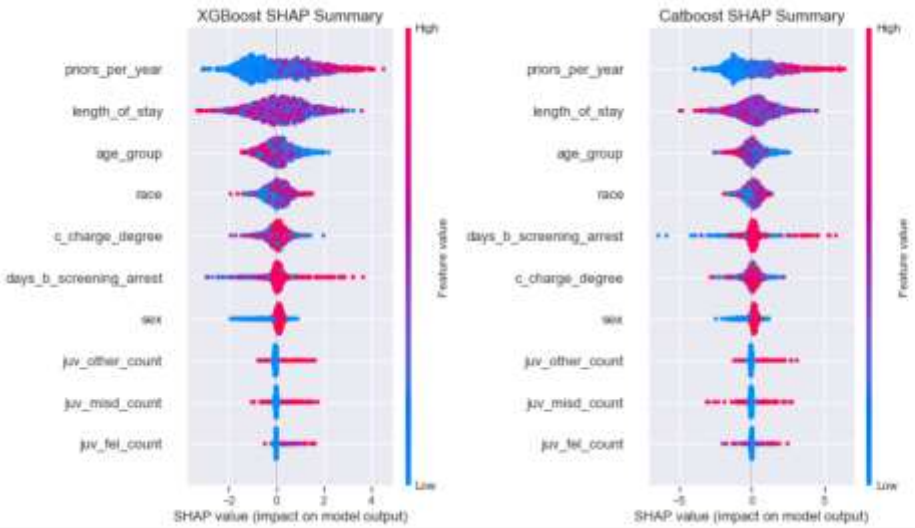

```
African-American FPR:   16.9%
Caucasian FPR:          16.1%
Ratio FPRs:             1.06 x
```

Examining Racial Bias through Feature Importance

```
fitted_cb_mdl = fitted_class_mdls['catboost_opt']['fitted']
shap_cb_explainer = shap.TreeExplainer(fitted_cb_mdl)
shap_cb_values = shap_cb_explainer.shap_values(X_test)

fitted_xgb_mdl = fitted_class_mdls['xgb_reg']['fitted']
shap_xgb_explainer = shap.TreeExplainer(fitted_xgb_mdl)
shap_xgb_values = shap_xgb_explainer.shap_values(X_test)
```

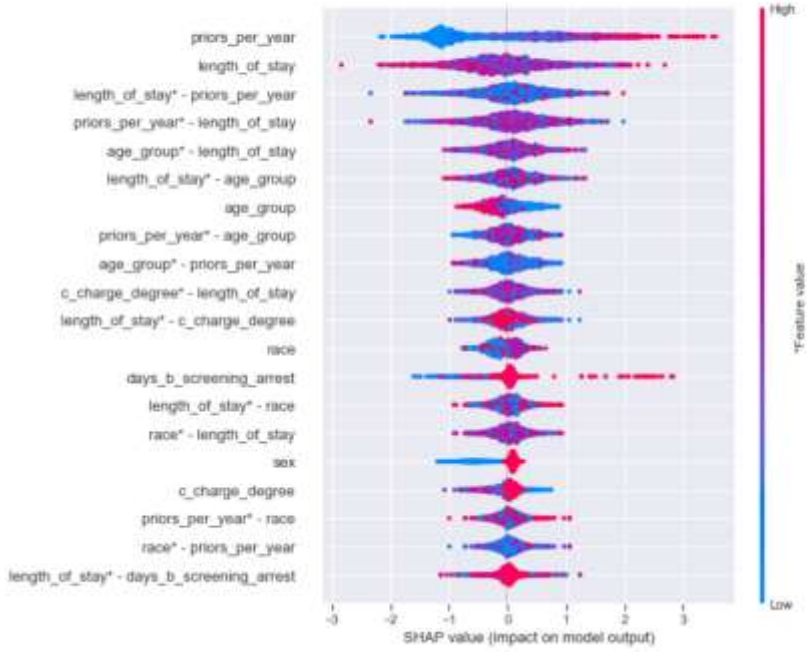
```
plt.figure(figsize=(12,7))

ax0 = plt.subplot(1, 2, 1)
shap.summary_plot(shap_xgb_values, X_test, plot_type="dot",\
                  plot_size=None, show=False)
ax0.set_title("XGBoost SHAP Summary", fontsize=15)
```

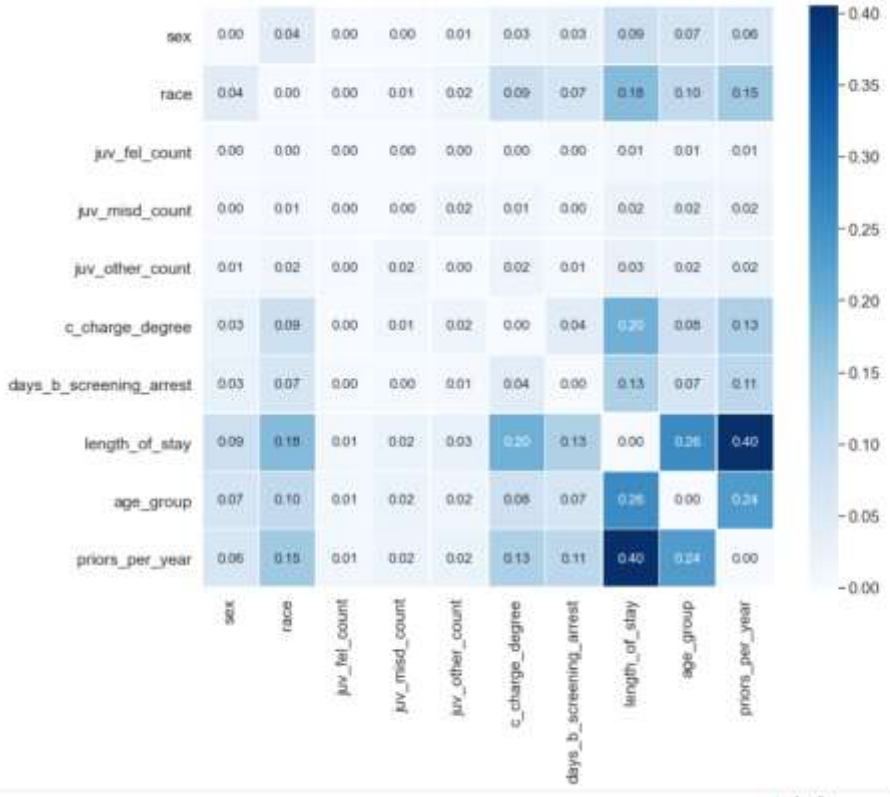
```
plt.figure(figsize=(12,7))

ax0 = plt.subplot(1, 2, 1)
shap.summary_plot(shap_xgb_values, X_test, plot_type="dot",\
                  plot_size=None, show=False)
ax0.set_title("XGBoost SHAP Summary", fontsize=15)

ax1 = plt.subplot(1, 2, 2)
shap.summary_plot(shap_cb_values, X_test, plot_type="dot",\
                  plot_size=None, show=False)
ax1.set_title("Catboost SHAP Summary", fontsize=15)

plt.tight_layout()
plt.subplots_adjust(wspace=0.8)
plt.show()
```



```
shap_xgb_interact_values = shap_xgb_explainer.shap_interaction_values(X_test)
```

```
shap.summary_plot(shap_xgb_interact_values, X_test,\
                  plot_type="compact_dot", sort=True)
```

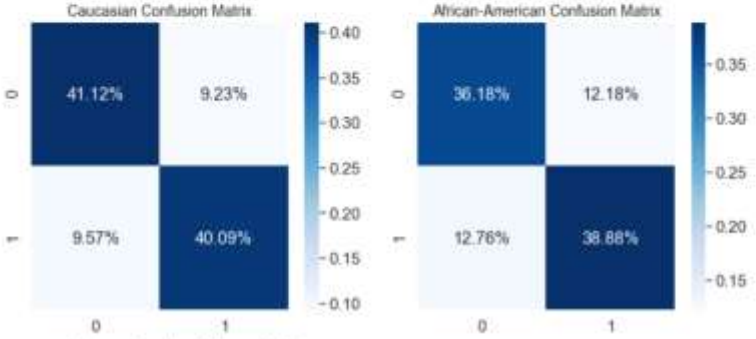




Training and Evaluating Constrained Model

```
xgb_con = xgb.XGBClassifier(seed=rand,\
                            monotone_constraints=mono_con,\
                            interaction_constraints=interact_con,\
                            **best_xgb_params)
xgb_con = xgb_con.fit(X_train_con, y_train)
fitted_class_mdls['xgb_con'] =\
        mldatasets.evaluate_class_mdl(xgb_con, X_train_con, X_test_con,\
                                      y_train, y_test, plot_roc=False,\
                                      ret_eval_dict=True)
y_test_pred = fitted_class_mdls['xgb_con']['preds_test']
_ = mldatasets.\
            compare_confusion_matrices(y_test[X_test.race==1], y_test_pred[X_test.race==1],\
                                       y_test[X_test.race==0], y_test_pred[X_test.race==0],\
                                       'Caucasian', 'African-American', compare_fpr=True)
```

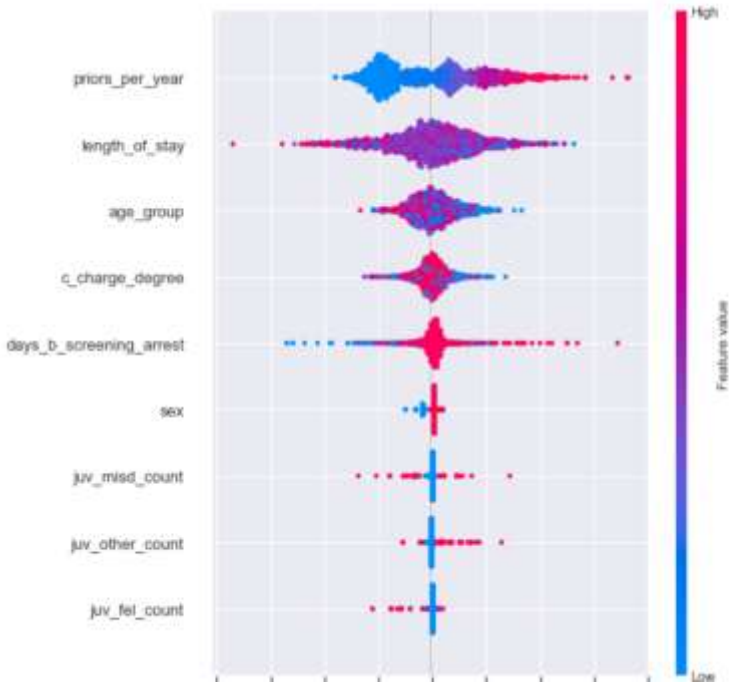
```
[13:28:59] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with t
The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1)
Accuracy_train:  0.9252          Accuracy_test:   0.7981
Precision_test:  0.7998          Recall_test:     0.8041
ROC-AUC_test:    0.8633          F1_test:         0.8019          MCC_test: 0.5961
```

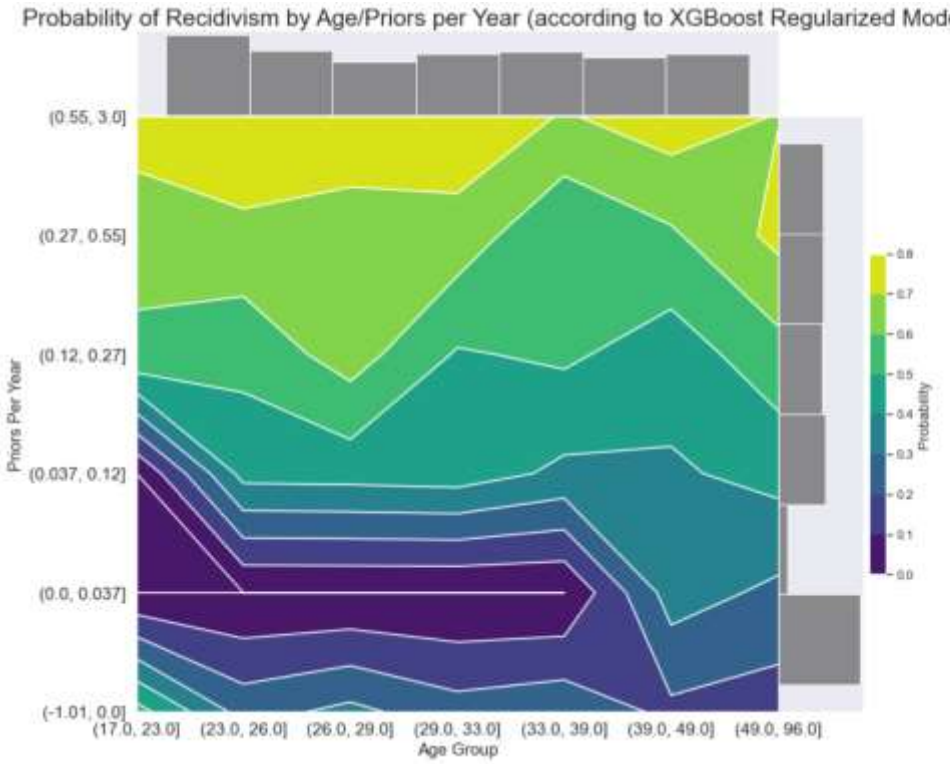


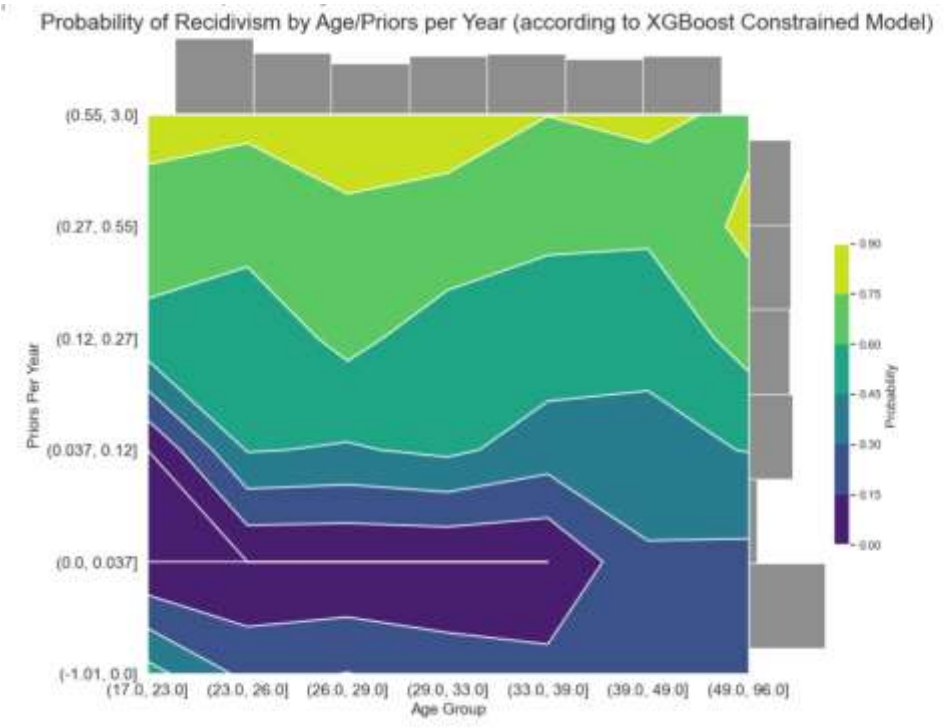

```
African-American FPR:   25.2%
```

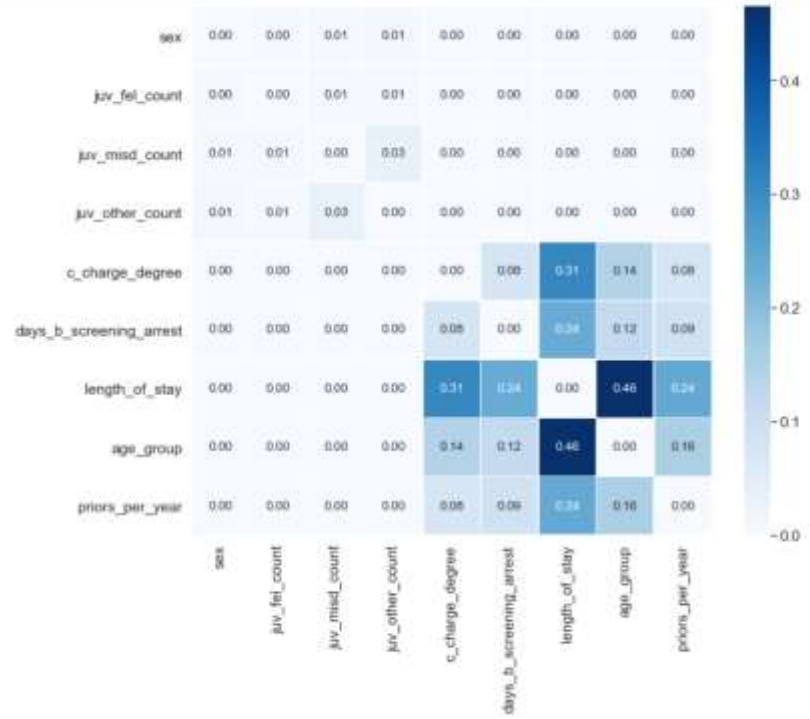
Examining Constraints

```
fitted_xgb_con_mdl = fitted_class_mdls['xgb_con']['fitted']
shap_xgb_con_explainer = shap.TreeExplainer(fitted_xgb_con_mdl)
shap_xgb_con_values = shap_xgb_con_explainer.shap_values(X_test_con)
shap.summary_plot(shap_xgb_con_values, X_test_con, plot_type="dot", show=False)
fig = plt.gcf()
fig.set_size_inches(8,10)
plt.show()
```

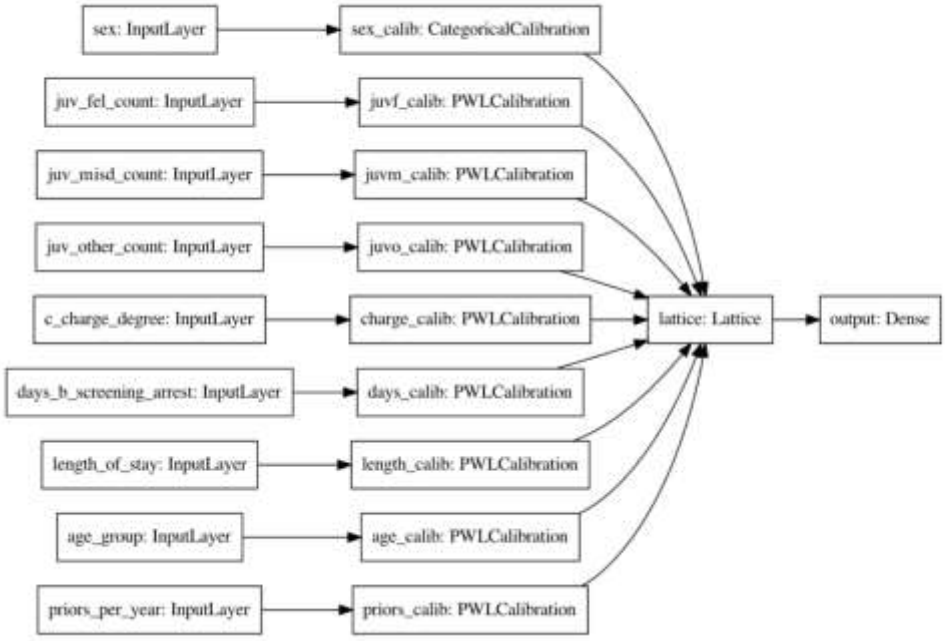



Probability of Recidivism by Age/Priors per Year (according to XGBoost Regularized Model)
(0.55, 3.0]
(0.27, 0.55]
(0.12, 0.27]
(0.037, 0.12]
(0.0, 0.037]
(-1.01, 0.0]
Priors Per Year
(17.0, 23.0]
(23.0, 26.0]
(26.0, 29.0]
(29.0, 33.0]
(33.0, 39.0]
(39.0, 49.0]
(49.0, 96.0]
Age Group
Probability
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0


Probability of Recidivism by Age/Priors per Year (according to XGBoost Constrained Model)
(0.55, 3.0]
(0.27, 0.55]
(0.12, 0.27]
(0.037, 0.12]
(0.0, 0.037]
(-1.01, 0.0]
Priors Per Year
(17.0, 23.0]
(23.0, 26.0]
(26.0, 29.0]
(29.0, 33.0]
(33.0, 39.0]
(39.0, 49.0]
(49.0, 96.0]
Age Group
Probability
0.90
0.75
0.60
0.45
0.30
0.15
0.00



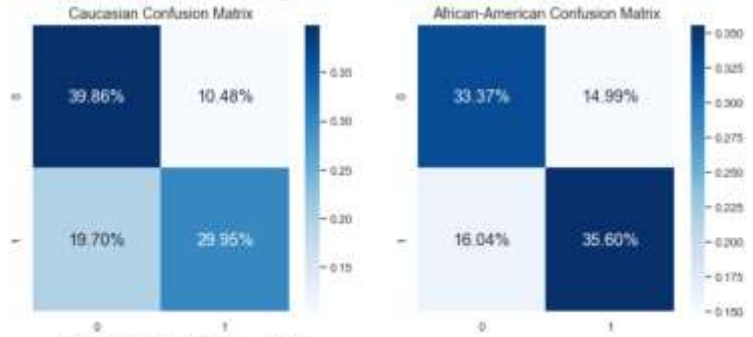

```
[ ] tf.keras.utils.plot_model(tfl_mdl, rankdir='LR')
```



+ Code + Text

```
fitted_class_mdls['tfl_con'] =\
    mldatasets.evaluate_class_mdl(tfl_mdl, X_train_expand,\
                                  X_test_expand,\
                                  y_train.values.astype(np.float32),\
                                  y_test.values.astype(np.float32),\
                                  plot_roc=False, ret_eval_dict=True)
y_test_pred = fitted_class_mdls['tfl_con']['preds_test']
_ = mldatasets.\
        compare_confusion_matrices(y_test[X_test.race==1], y_test_pred[X_test.race==1],\
                                   y_test[X_test.race==0], y_test_pred[X_test.race==0],\
                                   'Caucasian', 'African-American', compare_fpr=True)
```

```
Accuracy_train:  0.7453        Accuracy_test:   0.7057
Precision_test:  0.7387        Recall_test:     0.6514
ROC-AUC_test:    0.7651        F1_test:         0.6923        MCC_test: 0.4154
```




```
African-American FPR:  31.0%
Caucasian FPR:         20.8%
Ratio FPRs:            1.49 x
```

Mission Accomplished

```
for mdl_name in fitted_class_mdls:
    fitted_class_mdls[mdl_name]['wppra_test'] =\
            weighted_penalized_pr_average(y_test,\
                                          fitted_class_mdls[mdl_name]['preds_test'],\
                                          X_test['race'], range(3))

class_metrics = pd.DataFrame.from_dict(fitted_class_mdls,\
                    'index')[['precision_test', 'recall_test', 'wppra_test']]
with pd.option_context('display.precision', 3):
    html = class_metrics.sort_values(by='wppra_test', ascending=False).\
        style.background_gradient(cmap='plasma', subset=['precision_test']).\
        background_gradient(cmap='viridis', subset=['recall_test'])
html
```

	precision_test	recall_test	wppra_test
catboost_opt	0.835	0.824	0.815
catboost_reg	0.837	0.802	0.810
xgb_base	0.812	0.848	0.806
catboost_base	0.809	0.838	0.800
xgb_reg	0.828	0.815	0.797
nu-svc_reg	0.836	0.772	0.791
xgb_con	0.800	0.804	0.781
lgbm_reg	0.798	0.709	0.747
lgbm_base	0.766	0.748	0.743
xgb-rf_base	0.698	0.718	0.693
rf_reg	0.716	0.635	0.682
tfl_con	0.731	0.629	0.677
rf_base	0.687	0.686	0.669
xgb-rf_reg	0.703	0.643	0.668
mlp_base	0.689	0.596	0.639
mlp_reg	0.724	0.514	0.636
svc_reg	0.728	0.491	0.634
logistic_base	0.712	0.531	0.631
keras_reg	0.707	0.525	0.631
logistic_reg	0.745	0.437	0.627
svc_base	0.682	0.521	0.546

	precision_test	recall_test	wppra_test
catboost_opt	0.835	0.824	0.815
catboost_reg	0.837	0.802	0.810
xgb_base	0.812	0.848	0.806
catboost_base	0.809	0.838	0.800
xgb_reg	0.828	0.815	0.797
nu-svc_reg	0.836	0.772	0.791
xgb_con	0.800	0.804	0.781
lgbm_reg	0.798	0.709	0.747
lgbm_base	0.766	0.748	0.743
xgb-rf_base	0.698	0.718	0.693
rf_reg	0.716	0.635	0.682
tfl_con	0.731	0.629	0.677
rf_base	0.687	0.686	0.669
xgb-rf_reg	0.703	0.643	0.668
mlp_base	0.689	0.596	0.639
mlp_reg	0.724	0.514	0.636
svc_reg	0.728	0.491	0.634
logistic_base	0.712	0.531	0.631
keras_reg	0.707	0.525	0.631
logistic_reg	0.745	0.437	0.627
svc_base	0.583	0.523	0.546
nu-svc_base	0.580	0.122	0.406

Chapter 13 (Masks):

Part 1-

```
!pip install --upgrade numpy scikit-learn tensorflow matplotlib seaborn tqdm
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from tensorboard<2.12,>=2.11->tensorflow) (1.0.1)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.8/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow) (4.9)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow) (5.3.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.8/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow) (0.2.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.8/dist-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tensorflow) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.8/dist-packages (from markdown>=2.6.8->tensorboard<2.12,>=2.11->tensorflow) (6.0.0)
Requirement already satisfied: charset-normalizer<3,>=2.0.12 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (2.0.12)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (2022.12.7)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (1.24.3)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.8/dist-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.12,>=2.11->tensorflow) (3.12.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.8/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow) (0
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.8/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tenso
Installing collected packages: flatbuffers, tensorflow-estimator, numpy, keras, fasttcouls, contourpy, scikit-learn, matplotlib, tensorboard, seaborn, tensorflow
  Attempting uninstall: flatbuffers
    Found existing installation: flatbuffers 1.12
    Uninstalling flatbuffers-1.12:
      Successfully uninstalled flatbuffers-1.12
  Attempting uninstall: tensorflow-estimator
    Found existing installation: tensorflow-estimator 2.9.0
    Uninstalling tensorflow-estimator-2.9.0:
      Successfully uninstalled tensorflow-estimator-2.9.0
  Attempting uninstall: numpy
    Found existing installation: numpy 1.21.6
    Uninstalling numpy-1.21.6:
      Successfully uninstalled numpy-1.21.6
  Attempting uninstall: keras
    Found existing installation: keras 2.9.0
    Uninstalling keras-2.9.0:
      Successfully uninstalled keras-2.9.0
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.0.2
    Uninstalling scikit-learn-1.0.2:
```

```
!pip install --upgrade machine-learning-datasets
!pip install --upgrade adversarial-robustness-toolbox
```

```
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.8/dist-packages (from scikit-image!=0.17.1,<0.19,>=0.14.2->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (2.9.0)
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.8/dist-packages (from scikit-image!=0.17.1,<0.19,>=0.14.2->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (2021.11.2)
Requirement already satisfied: typer<0.8.0,>=0.3.0 in /usr/local/lib/python3.8/dist-packages (from spacy[lookups]<4.0.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (0.7.0)
Requirement already satisfied: pathy>=0.3.5 in /usr/local/lib/python3.8/dist-packages (from spacy[lookups]<4.0.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (0.10.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.8/dist-packages (from spacy[lookups]<4.0.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (2.11.3)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.8/dist-packages (from spacy[lookups]<4.0.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (3.3.0)
Requirement already satisfied: wasabi<1.1.0,>=0.9.1 in /usr/local/lib/python3.8/dist-packages (from spacy[lookups]<4.0.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (0.10.1)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.8/dist-packages (from spacy[lookups]<4.0.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (2.0.7)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.10 in /usr/local/lib/python3.8/dist-packages (from spacy[lookups]<4.0.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (3.0.10)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.8/dist-packages (from spacy[lookups]<4.0.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (2.0.8)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from spacy[lookups]<4.0.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (3.0.8)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.8/dist-packages (from spacy[lookups]<4.0.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (1.0.4)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /usr/local/lib/python3.8/dist-packages (from spacy[lookups]<4.0.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (6.3.0)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.8/dist-packages (from spacy[lookups]<4.0.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (2.4.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.8/dist-packages (from spacy[lookups]<4.0.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (1.0.9)
Requirement already satisfied: pydantic!<1.8.1,!<1.8.1,<1.11.0,>=1.7.4 in /usr/local/lib/python3.8/dist-packages (from spacy[lookups]<4.0.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (1.10.2)
Requirement already satisfied: thinc<8.2.0,>=8.1.0 in /usr/local/lib/python3.8/dist-packages (from spacy[lookups]<4.0.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (8.1.7)
Collecting spacy-lookups-data<1.1.0,>=1.0.5
  Downloading spacy_lookups_data-1.0.3-py2.py3-none-any.whl (98.5 MB)
     ---------------------------------------- 98.5/98.5 MB 9.1 MB/s eta 0:00:00
Requirement already satisfied: astunparse~=1.6.3 in /usr/local/lib/python3.8/dist-packages (from tensorflow<2.5.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (1.6.3)
Requirement already satisfied: keras-preprocessing~=1.1.2 in /usr/local/lib/python3.8/dist-packages (from tensorflow<2.5.0,>=2.0.0->alibi>0.6.0,>=0.5.5->machine-learning-datasets) (1.1.2)
Collecting termcolor~=1.1.0
  Downloading termcolor-1.1.0.tar.gz (3.9 kB)
  Preparing metadata (setup.py) ... done
Collecting grpcio~=1.32.0
  Downloading grpcio-1.32.0-cp38-cp38-manylinux2014_x86_64.whl (3.8 MB)
     ---------------------------------------- 3.8/3.8 MB 66.0 MB/s eta 0:00:00
Collecting flatbuffers~=1.12.0
  Downloading flatbuffers-1.12-py2.py3-none-any.whl (15 kB)
Collecting wrapt~=1.12.1
```

```
import math
import os
import warnings
warnings.filterwarnings("ignore")
import machine_learning_datasets as mldatasets
import numpy as np
from sklearn import preprocessing
import tensorflow as tf
from tensorflow.keras.utils import get_file
import matplotlib.pyplot as plt
import seaborn as sns

#PART 1 only
from sklearn import metrics
from art.estimators.classification import KerasClassifier
from art.attacks.evasion import FastGradientMethod, ProjectedGradientDescent, BasicIterativeMethod
from art.attacks.evasion import CarliniLInfMethod
from art.attacks.evasion import AdversarialPatchNumpy
from art.defences.preprocessor import SpatialSmoothing
from art.defences.trainer import AdversarialTrainer
from tqdm.notebook import tqdm

#PART 2 only
from art.estimators.classification import TensorFlowV2Classifier
from art.estimators.certification.randomized_smoothing import TensorFlowV2RandomizedSmoothing
from art.utils import compute_accuracy
```

```
print(tf.__version__)
```

```
2.2.1
```

```
tf.compat.v1.disable_eager_execution()
print('Eager execution enabled: ', tf.executing_eagerly())
```

```
Eager execution enabled:  False
```

Understanding and Preparing the Data

```
X_train, X_test, y_train, y_test =\
                                mldatasets.load("masksface-net_thumbs_sampled", preprocess=True)
X_train, X_test = X_train / 255.0, X_test / 255.0
min_ = X_train.min()
max_ = X_train.max()
```

```
https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python/raw/master/datasets/masksface-net_thumbs_sampled.zip downloaded to /Users/smasis/Documents/OTHER/InterpretableML
/Users/smasis/Documents/OTHER/Interpretable-Book/programming/Chapter13/data/masksface-net_thumbs_sampled.zip uncompressed to /Users/smasis/Documents/OTHER/Interpretable-Book/programming/Cha
21000 dataset files found in /Users/smasis/Documents/OTHER/Interpretable-Book/programming/Chapter13/data/masksface-net_thumbs_sampled folder
```

```
print('X_train dim:\t%s' % (X_train.shape,))
print('X_test dim:\t%s' % (X_test.shape,))
print('y_train dim:\t%s' % (y_train.shape,))
print('y_test dim:\t%s' % (y_test.shape,))
print('X_train min:\t%s' % (min_))
print('X_train max:\t%s' % (max_))
print('y_train labels:\t%s' % (np.unique(y_train)))
```

```
X_train dim:    (16000, 128, 128, 3)
X_test dim:     (4200, 128, 128, 3)
y_train dim:    (16000, 1)
y_test dim:     (4200, 1)
X_train min:    0.0
```

```
X_train min:    0.0
X_train max:    1.0
y_train labels: ['Correct' 'Incorrect' 'None']
```

```
ohe = preprocessing.OneHotEncoder(sparse=False)
ohe.fit(y_train)
labels_1 = ohe.categories_[0].tolist()
print(labels_1)
```

```
['Correct', 'Incorrect', 'None']
```

```
rand = 9
os.environ['PYTHONHASHSEED'] = str(rand)
tf.random.set_seed(rand)
np.random.seed(rand)
```

```
sampl_md_idxs = np.random.choice(X_test.shape[0], 200, replace=False)
X_test_mdsample = X_test[sampl_md_idxs]
y_test_mdsample = y_test[sampl_md_idxs]
sampl_sm_idxs = np.random.choice(X_test.shape[0], 20, replace=False)
X_test_smsample = X_test[sampl_sm_idxs]
y_test_smsample = y_test[sampl_sm_idxs]
```

```
plt.subplots(figsize=(15,12))
for s in range(20):
    plt.subplot(4, 5, s+1)
    plt.title(y_test_smsample[s][0], fontsize=12)
    plt.imshow(X_test_smsample[s], interpolation='spline16')
    plt.axis('off')
plt.show()
```

Training the CNN Base Model

Optional

```
base_model = tf.keras.models.Sequential([
    tf.keras.layers.InputLayer(input_shape=X_train.shape[1:]),
    tf.keras.layers.Conv2D(16, kernel_size=(3, 3), activation='relu', name='conv2d_1'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='maxpool2d_1'),
    tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu', name='conv2d_2'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='maxpool2d_2'),
    tf.keras.layers.Conv2D(64, kernel_size=(3, 3), activation='relu', name='conv2d_3'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='maxpool2d_3'),
    tf.keras.layers.Conv2D(128, kernel_size=(3, 3), activation='relu', name='conv2d_4'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='maxpool2d_4'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(768, activation='relu', name='dense_1'),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Dense(3, activation='softmax', name='dense_2')
], name='CNN_Base_MaskedFaceNet_Model')

base_model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001),
                   loss='categorical_crossentropy',
                   metrics=['accuracy'])

base_model.summary()
```

```
WARNING:tensorflow:From /opt/anaconda3/lib/python3.8/site-packages/tensorflow/python/ops/resource_variable_ops.py:1659: calling BaseResourceVariable.__init__ (from tensorf
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
Model: "CNN_Base_MaskedFaceNet_Model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 126, 126, 16)      448
_________________________________________________________________
maxpool2d_1 (MaxPooling2D)   (None, 63, 63, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 61, 61, 32)        4640
_________________________________________________________________
maxpool2d_2 (MaxPooling2D)   (None, 30, 30, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 64)        18496
_________________________________________________________________
maxpool2d_3 (MaxPooling2D)   (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 128)       73856
_________________________________________________________________
maxpool2d_4 (MaxPooling2D)   (None, 6, 6, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense_1 (Dense)              (None, 768)               3539712
_________________________________________________________________
dropout (Dropout)            (None, 768)               0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 2307
=================================================================
Total params: 3,639,455
Trainable params: 3,639,459
Non-trainable params: 0
_________________________________________________________________
```

```
[ ] es = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', mode='max', verbose=1,\
                                          patience=3, restore_best_weights=True)
```

```
[ ] es = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', mode='max', verbose=1,\
                                          patience=3, restore_best_weights=True)

    base_model.fit(X_train, ohe.transform(y_train), validation_split=0.2,\
                   batch_size=128, callbacks=[es], epochs=60, verbose=1)

    _, accuracy_test = base_model.evaluate(X_test, ohe.transform(y_test))
    print('Accuracy on test data: {:4.2f}%'.format(accuracy_test * 100))
```

Show hidden output

Loading the CNN Base Model

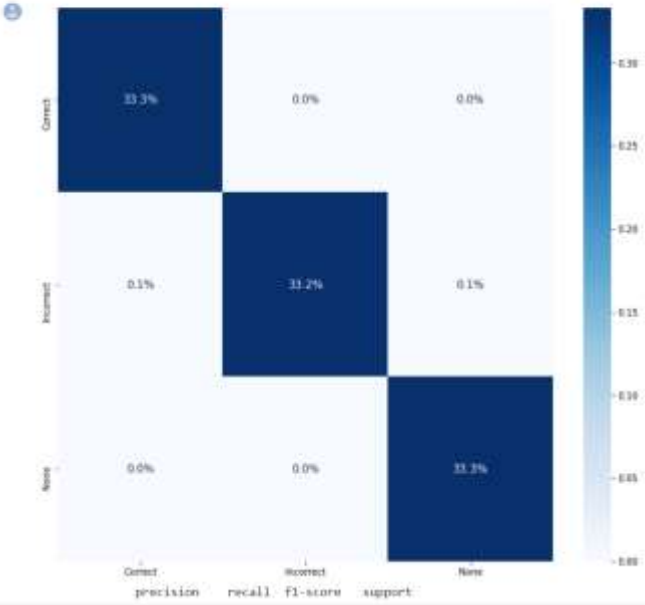
```
model_path = get_file('CNN_Base_MaskedFace_Net.hdf5',\
                      'https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python/blob/master/models/CNN_Base_MaskedFace_Net.hdf5?raw=true')
base_model = tf.keras.models.load_model(model_path)
base_model.summary()
```

```
Downloading data from https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python/blob/master/models/CNN_Base_MaskedFace_Net.hdf5?raw=true
43737088/43734584 [==============================] - 0s 0us/step
Model: "CNN_Base_MaskedFaceNet_Model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 126, 126, 16)      448
_________________________________________________________________
maxpool2d_1 (MaxPooling2D)   (None, 63, 63, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 61, 61, 32)        4640
_________________________________________________________________
maxpool2d_2 (MaxPooling2D)   (None, 30, 30, 32)        0
_________________________________________________________________
```

```
conv2d_1 (Conv2D)            (None, 126, 126, 16)      448
maxpool2d_1 (MaxPooling2D)   (None, 63, 63, 16)        0
conv2d_2 (Conv2D)            (None, 61, 61, 32)        4640
maxpool2d_2 (MaxPooling2D)   (None, 30, 30, 32)        0
conv2d_3 (Conv2D)            (None, 28, 28, 64)        18496
maxpool2d_3 (MaxPooling2D)   (None, 14, 14, 64)        0
conv2d_4 (Conv2D)            (None, 12, 12, 128)       73856
maxpool2d_4 (MaxPooling2D)   (None, 6, 6, 128)         0
flatten_6 (Flatten)          (None, 4608)              0
dense_1 (Dense)              (None, 768)               3539712
dropout_6 (Dropout)          (None, 768)               0
dense_2 (Dense)              (None, 3)                 2307
=================================================================
Total params: 3,639,459
Trainable params: 3,639,459
Non-trainable params: 0
```

Assessing the CNN Base Classifier

```
y_test_pred, y_test_prob = mldatasets.evaluate_multiclass_mdl(base_model, X_test, y_test, labels_l, ohe,\
                                                              plot_conf_matrix=True, predopts={"verbose":1})
```




```
              precision    recall  f1-score   support

     Correct       0.998     0.998     0.998      1400
   Incorrect       0.998     0.995     0.997      1400
        None       0.996     0.999     0.998      1400

    accuracy                           0.997      4200
   macro avg       0.997     0.997     0.997      4200
weighted avg       0.997     0.997     0.997      4200
```

Learning about Evasion Attacks

```
base_classifier = KerasClassifier(model=base_model, clip_values=(min_, max_))
y_test_mdsample_prob = np.asarray(y_test_prob[sample_md_idxs], axis=1)
y_test_mmsample_prob = np.asarray(y_test_prob[sample_mm_idxs], axis=1)
```

Fast Gradient Sign Method Attack

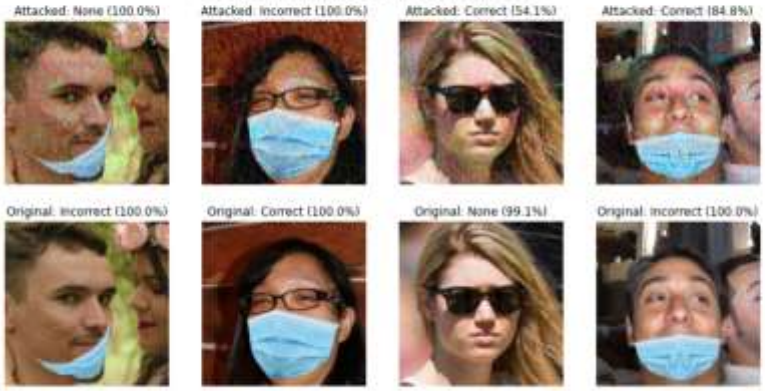
```
attack_fgsm = FastGradientMethod(base_classifier, eps=0.1)
```

```
X_test_fgsm = attack_fgsm.generate(X_test_mdsample)
```

```
y_test_fgsm_pred, y_test_fgsm_prob =\
    mldatasets.evaluate_multiclass_mdl(base_classifier.model, X_test_fgsm, y_test_mdsample,\
                                       labels_l, ohe, plot_conf_matrix=False, plot_roc=False)
y_test_fgsm_prob = np.max(y_test_fgsm_prob, axis=1)

mldatasets.compare_image_predictions(X_test_fgsm, X_test_mdsample, y_test_fgsm_pred,\
                                     y_test_mdsample.flatten(), y_test_fgsm_prob,\
                                     y_test_mdsample_prob, title_mod_prefix="Attacked: ",\
                                     title_difference_prefix="FGSM Attack Average Perturbation: ",\
                                     num_samples=4)
```

Correct	0.056	0.014	0.023	70
Incorrect	0.220	0.370	0.276	54
None	0.736	0.882	0.802	76
accuracy			0.440	200
macro avg	0.337	0.422	0.367	200
weighted avg	0.359	0.440	0.387	200

FSGM Attack Average Perturbation: 0.092



Carlini & Wagner Infinity-norm Attack

Carlini & Wagner Infinity-norm Attack

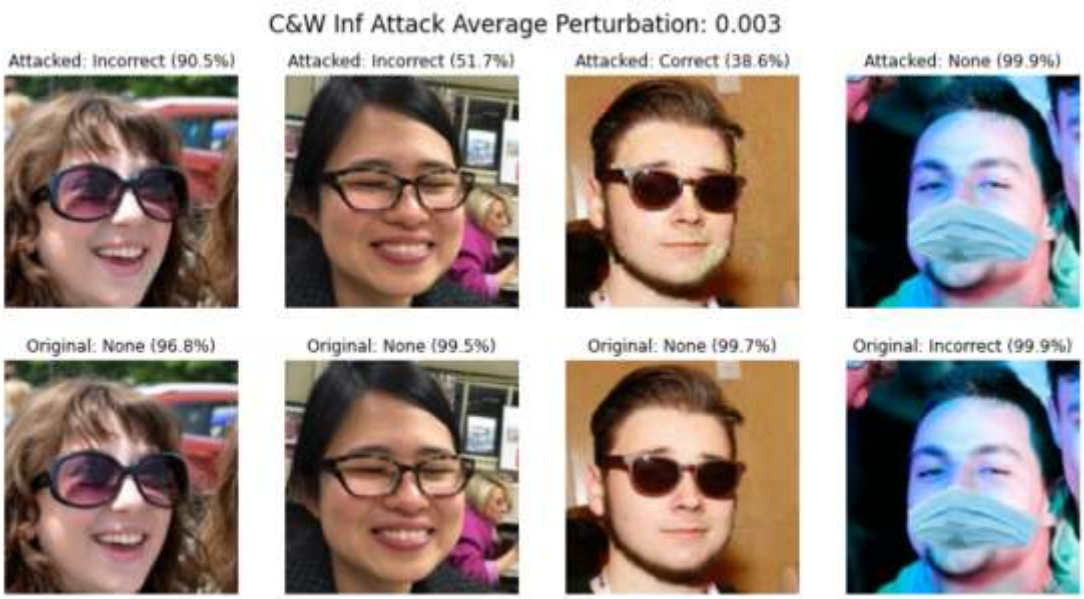
```
attack_cw = CarliniLInfMethod(base_classifier, eps=0.3, batch_size=40)
X_test_cw = attack_cw.generate(X_test_mdsample)
```

```
C&W L_inf: 100%|██████████| 5/5 [01:01<00:00, 12.32s/it]
```

```
y_test_cw_pred, y_test_cw_prob =\
    mldatasets.evaluate_multiclass_nn(base_classifier.model, X_test_cw, y_test_mdsample,\
                                      labels_1, ohe, plot_conf_matrix=False, plot_roc=False)
y_test_cw_prob = np.max(y_test_cw_prob, axis=1)

mldatasets.compare_image_predictions(X_test_cw, X_test_mdsample, y_test_cw_pred,\
                                     y_test_mdsample.flatten(), y_test_cw_prob,\
                                     y_test_mdsample_prob, title_mod_prefix="Attacked: ",\
                                     title_difference_prefix="C&W Inf Attack Average Perturbation: ",\
                                     num_samples=4)
```

	precision	recall	f1-score	support
Correct	0.932	0.971	0.951	70
Incorrect	0.839	0.963	0.897	54
None	0.985	0.842	0.908	76
accuracy			0.930	200
macro avg	0.918	0.925	0.918	200
weighted avg	0.927	0.920	0.920	200

C&W Inf Attack Average Perturbation: 0.003



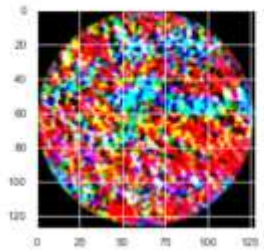
Targeted Adversarial Patch Attack

```
[ ] attack_ap = AdversarialPatchNumpy(base_classifier, scale_min=0.4, scale_max=0.7,\
                                      learning_rate=5., max_iter=500, batch_size=40,\
                                      target=0)
```

```
[ ] placement_mask = np.zeros((128,128))
    placement_mask[80:93,45:83] = 1
    placement_mask = np.expand_dims(placement_mask, axis=0).astype(bool)

    patch, patch_mask = attack_ap.generate(x=X_test_smsample,\
                                           y=ohe.transform(y_test_smsample),\
                                           mask=placement_mask)
```

```
    Adversarial Patch Numpy: 100%|██████████| 500/500 [05:39<00:00,  2.28it/s]
```

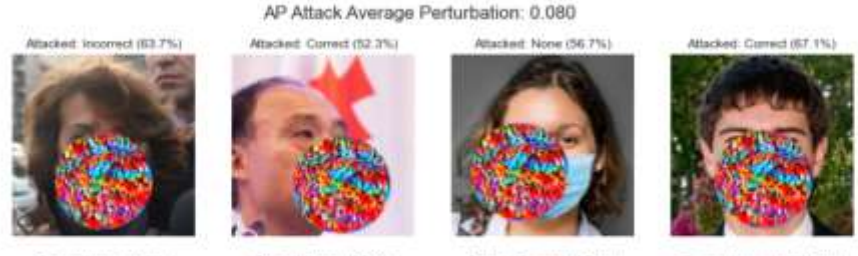
```
[ ] plt.imshow(patch * patch_mask)
    plt.show()
```



```
[ ] X_test_ap = attack_ap.apply_patch(X_test_smsample, scale=0.55, mask=placement_mask)
```

```
y_test_ap_pred, y_test_ap_prob =\
    mldatasets.evaluate_multiclass_mdl(base_classifier.model, X_test_ap, y_test_smsample,\
                                       labels_1, ohe, plot_conf_matrix=False, plot_roc=False)
y_test_ap_prob = np.max(y_test_ap_prob, axis=1)

mldatasets.compare_image_predictions(X_test_ap, X_test_smsample, y_test_ap_pred,\
                                     y_test_smsample.flatten(), y_test_ap_prob,\
                                     y_test_smsample_prob, title_mod_prefix="Attacked: ",\
                                     title_difference_prefix="AP Attack Average Perturbation: ",\
                                     num_samples=4)
```

```
              precision    recall  f1-score   support

     Correct      0.667     0.571     0.615         7
   Incorrect      0.750     0.600     0.667         5
        None      0.600     0.750     0.667         8

    accuracy                          0.650        20
   macro avg      0.672     0.640     0.650        20
weighted avg      0.661     0.650     0.649        20
```




```
[ ]
```

Defending Against Targeted Attacks with Preprocessing

+ Code + Text

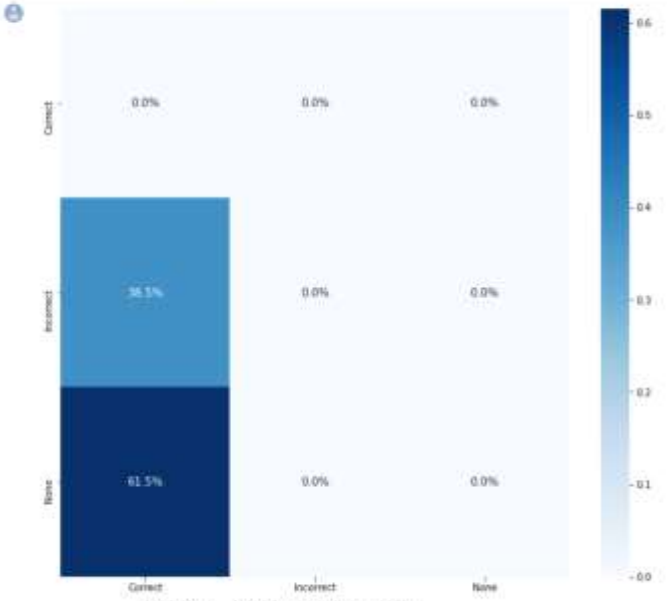
```
[ ] not_masked_idxs = np.where(y_test_smsample != 'Correct')[0]
    X_test_notmasked = X_test_smsample[not_masked_idxs]
    y_test_notmasked = y_test_smsample[not_masked_idxs]
    y_test_notmasked_prob = y_test_smsample_prob[not_masked_idxs]
    y_test_masked = np.array(['Correct'] * X_test_notmasked.shape[0]).reshape(-1,1)
```

```
[ ] attack_pgd = ProjectedGradientDescent(base_classifier, eps=0.3, eps_step=0.01, max_iter=40,\
                                          targeted=True)
    X_test_pgd = attack_pgd.generate(X_test_notmasked,\
                                     y=ohe.transform(y_test_masked))
```

Show hidden output

```
[ ] y_test_pgd_pred, y_test_pgd_prob =\
        mldatasets.evaluate_multiclass_mdl(base_classifier.model, X_test_pgd, y_test_notmasked,\
                                           labels_1, ohe, plot_conf_matrix=True, plot_roc=False)
    y_test_pgd_prob = np.max(y_test_pgd_prob, axis=1)
```

```
y_test_pgd_pred, y_test_pgd_prob =\
    mldatasets.evaluate_multiclass_mdl(base_classifier.model, X_test_pgd, y_test_notmasked,\
                                       labels_1, ohe, plot_conf_matrix=True, plot_roc=False)
y_test_pgd_prob = np.max(y_test_pgd_prob, axis=1)
```




```
              precision    recall  f1-score   support

     Correct      0.000     0.000     0.000       0.0
```

```
mldatasets.compare_image_predictions(X_test_pgd, X_test_notmasked, y_test_pgd_pred,\
                                     y_test_notmasked.flatten(), y_test_pgd_prob, y_test_samsample_prob,\
                                     title_mod_prefix="Attacked: ", num_samples=4,\
                                     title_difference_prefix="PGD Attack Average Perturbation: ")
```


```
[ ] defence_ss = SpatialSmoothing(window_size=11)
    X_test_pgd_ss, _ = defence_ss(X_test_pgd)
```

```
[ ] y_test_pgd_ss_pred, y_test_pgd_ss_prob =\
        mldatasets.evaluate_multiclass_mdl(base_classifier.model, X_test_pgd_ss, y_test_notmasked,\
                                           labels_1, ohe, plot_conf_matrix=False, plot_roc=False)
    y_test_pgd_ss_prob = np.max(y_test_pgd_ss_prob, axis=1)
```

```
              precision    recall  f1-score   support

     Correct      0.000     0.000     0.000         0
   Incorrect      1.000     0.600     0.750         5
```

```
mldatasets.compare_image_predictions(X_test_pgd_ss, X_test_notmasked, y_test_pgd_ss_pred,\
                                     y_test_notmasked.flatten(), y_test_pgd_ss_prob, y_test_notmasked_prob,\
                                     title_pred_prefix="Attacked+Defended:", num_samples=4,\
                                     title_difference_prefix="PGD Attack & Defended Average Perturbation:",\
                                     use_misclass=False)
```

Shielding Against Any Evasion Attack by Adversarial Training of a Robust Classifier

Training the CNN Robust Model

Optional

```
robust_model = tf.keras.models.Sequential([
    tf.keras.layers.InputLayer(input_shape=X_train.shape[1:]),
    tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu', name='conv2d_1'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='maxpool2d_1'),
    tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu', name='conv2d_2'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='maxpool2d_2'),
    tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu', name='conv2d_3'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='maxpool2d_3'),
    tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu', name='conv2d_4'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='maxpool2d_4'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(3072, activation='relu', name='dense_1'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(3, activation='softmax', name='dense_2')
], name='CNN_Robust_MaskedFaceNet_Model')

robust_model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001),
                     loss='categorical_crossentropy',
                     metrics=['accuracy'])

robust_model.summary()
```

```
Model: "CNN_Robust_MaskedFaceNet_Model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 126, 126, 32)      896
_________________________________________________________________
maxpool2d_1 (MaxPooling2D)   (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 61, 61, 32)        9248
_________________________________________________________________
maxpool2d_2 (MaxPooling2D)   (None, 30, 30, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 32)        9248
_________________________________________________________________
maxpool2d_3 (MaxPooling2D)   (None, 14, 14, 32)        0
```

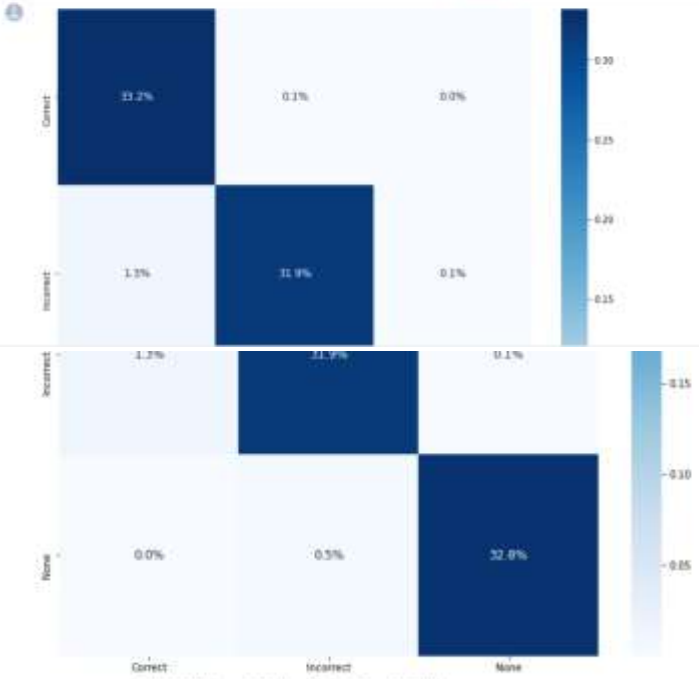
Loading the CNN Robust Model

```
[ ] model_path = get_file('CNN_Robust_MaskedFace_Net.hdf5',\
                          'https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python/blob/master/models/CNN_Robust_MaskedFace_Net.hdf5?raw=true')
    robust_model = tf.keras.models.load_model(model_path)
    robust_classifier = KerasClassifier(model=robust_model, clip_values=(min_, max_))
```

```
Downloading data from https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python/blob/master/models/CNN_Robust_MaskedFace_Net.hdf5?raw=true
43024384/43019888 [==============================] - 0s 0us/step
```

Assessing the CNN Robust Classifier

```
y_test_robust_pred, y_test_robust_prob =\
                mldatasets.evaluate_multiclass_mdl(robust_classifier.model, X_test, y_test,\
                                                   labels_l, ohe, plot_conf_matrix=True,\
                                                   predopts={"verbose":1})
```




```
              precision    recall  f1-score   support

     Correct       0.962     0.996     0.979      1400
   Incorrect       0.980     0.957     0.969      1400
        None       0.994     0.983     0.989      1400

    accuracy                           0.979      4200
   macro avg       0.979     0.979     0.979      4200
weighted avg       0.979     0.979     0.979      4200
```

```
[ ] attack_fgsm_robust = FastGradientMethod(robust_classifier, eps=0.1)
    X_test_fgsm_robust = attack_fgsm_robust.generate(X_test_mdsample)
```

```
[ ] y_test_fgsm_robust_pred, y_test_fgsm_robust_prob =\
        mldatasets.evaluate_multiclass_mdl(robust_classifier.model, X_test_fgsm_robust, y_test_mdsample,\
                                           labels_l, ohe, plot_conf_matrix=False, plot_roc=False)
    y_test_fgsm_robust_prob = np.max(y_test_fgsm_robust_prob, axis=1)
```

```
                                    labels_1, one, plot_conf_matrix=False, plot_roc=False)
y_test_fgsm_robust_prob = np.max(y_test_fgsm_robust_prob, axis=1)

mldatasets.compare_image_predictions(X_test_fgsm_robust, X_test_mdsample, y_test_fgsm_robust_pred,\
                                     y_test_mdsample.flatten(), y_test_fgsm_robust_prob,\
                                     y_test_mdsample_prob, title_mod_prefix="Attacked: ",\
                                     title_difference_prefix="FSGM Attack Average Perturbation: ",\
                                     num_samples=4)
```

```
              precision    recall  f1-score   support

     Correct      0.921     1.000     0.959        70
   Incorrect      0.942     0.907     0.925        54
        None      1.000     0.947     0.973        76

    accuracy                          0.955       200
   macro avg      0.954     0.952     0.952       200
weighted avg      0.957     0.955     0.955       200
```

Evaluating and Certifying Adversarial Robustness

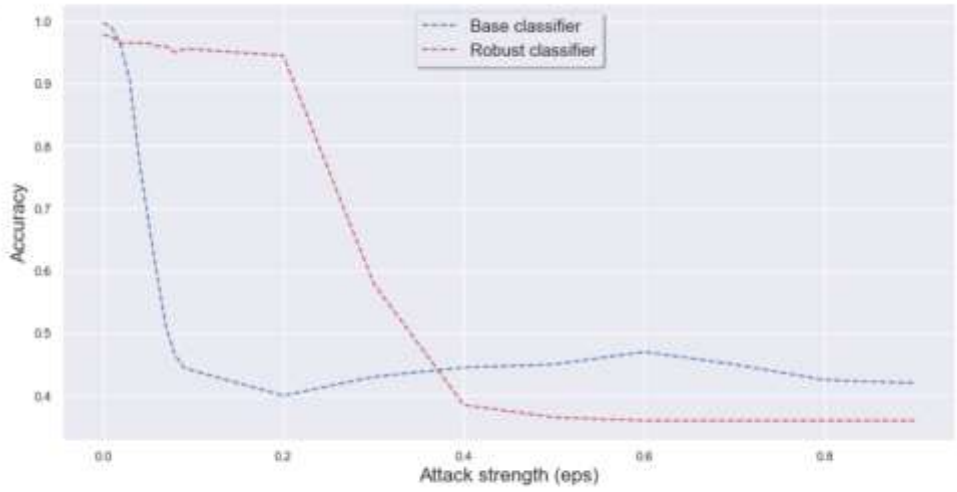
Comparing Model Robustness with Attack Strength

```
accuracy_base_0 = metrics.accuracy_score(y_test, y_test_pred)
accuracy_robust_0 = metrics.accuracy_score(y_test, y_test_robust_pred)
```

```
eps_range = np.concatenate((np.linspace(0.01, 0.09, 9),\
                            np.linspace(0.1, 0.9, 9)), axis=0).tolist()

accuracy_base = [accuracy_base_0]
accuracy_robust = [accuracy_robust_0]
for eps in tqdm(eps_range, desc='EPS'):
    attack_fgsm.set_params(**{'eps': eps})
    X_test_fgsm_base_i = attack_fgsm.generate(X_test_mdsample)
    _, accuracy_base_i = base_classifier.model.evaluate(X_test_fgsm_base_i,\
                                        ohe.transform(y_test_mdsample))

    attack_fgsm_robust.set_params(**{'eps': eps})
    X_test_fgsm_robust_i = attack_fgsm_robust.generate(X_test_mdsample)
    _, accuracy_robust_i = robust_classifier.model.evaluate(X_test_fgsm_robust_i,\
                                        ohe.transform(y_test_mdsample))

    accuracy_base.append(accuracy_base_i)
    accuracy_robust.append(accuracy_robust_i)

eps_range = [0] + eps_range
```

```
HBox(children=(HTML(value='EPS'), FloatProgress(value=0.0, max=18.0), HTML(value='')))

CPU times: user 5min 31s, sys: 41.2 s, total: 6min 12s
Wall time: 1min 13s
```

```
sns.set()
fig, ax = plt.subplots(figsize=(14,7))
ax.plot(np.array(eps_range), np.array(accuracy_base), 'b--', label='Base classifier')
ax.plot(np.array(eps_range), np.array(accuracy_robust), 'r--', label='Robust classifier')
```



Part 2:

```
!pip install --upgrade numpy scikit-learn tensorflow matplotlib seaborn tqdm

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (1.21.6)
Collecting numpy
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 32.1 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.8/dist-packages (1.0.2)
Collecting scikit-learn
  Downloading scikit_learn-1.2.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (9.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.8/9.8 MB 53.2 MB/s eta 0:00:00
Requirement already satisfied: tensorflow in /usr/local/lib/python3.8/dist-packages (2.9.2)
Collecting tensorflow
  Downloading tensorflow-2.11.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (588.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 588.3/588.3 MB 2.1 MB/s eta 0:00:00
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (3.2.2)
Collecting matplotlib
  Downloading matplotlib-3.6.3-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (9.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.4/9.4 MB 44.8 MB/s eta 0:00:00
Requirement already satisfied: seaborn in /usr/local/lib/python3.8/dist-packages (0.11.2)
Collecting seaborn
  Downloading seaborn-0.12.2-py3-none-any.whl (293 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 293.3/293.3 kB 19.8 MB/s eta 0:00:00
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (4.64.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (3.1.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (1.15.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.8/dist-packages (from tensorflow) (57.4.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (3.3.0)
Collecting keras<2.12,>=2.11.0
  Downloading keras-2.11.0-py2.py3-none-any.whl (1.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 68.8 MB/s eta 0:00:00
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (4.4.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (1.4.0)
Collecting tensorflow-estimator<2.12,>=2.11.0
  Downloading tensorflow_estimator-2.11.0-py2.py3-none-any.whl (439 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 439.2/439.2 kB 42.4 MB/s eta 0:00:00
Requirement already satisfied: packaging in /usr/local/lib/python3.8/dist-packages (from tensorflow) (23.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (3.1.0)
Collecting tensorboard<2.12,>=2.11
```

```
!pip install --upgrade machine-learning-datasets
!pip install --upgrade adversarial-robustness-toolbox
```

```
  Downloading tensorflow_estimator-2.6.0-py2.py3-none-any.whl (462 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 462.9/462.9 kB 31.2 MB/s eta 0:00:00
Collecting typing-extensions~=3.7.4
  Downloading typing_extensions-3.7.4.3-py3-none-any.whl (22 kB)
Requirement already satisfied: opt-einsum~=3.3.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (3.3.0)
Collecting wrapt~=1.12.1
  Downloading wrapt-1.12.1.tar.gz (27 kB)
  Preparing metadata (setup.py) ... done
Collecting h5py~=2.10.0
  Downloading h5py-2.10.0-cp38-cp38-manylinux1_x86_64.whl (2.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.9/2.9 MB 76.4 MB/s eta 0:00:00
Collecting absl-py~=0.10
  Downloading absl_py-0.15.0-py3-none-any.whl (132 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.0/132.0 KB 17.9 MB/s eta 0:00:00
Collecting numpy~=1.19.5
  Downloading numpy-1.19.5-cp38-cp38-manylinux2010_x86_64.whl (14.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.9/14.9 MB 76.0 MB/s eta 0:00:00
Requirement already satisfied: tensorboard~=2.4 in /usr/local/lib/python3.8/dist-packages (from tensorflow>1.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (2.11.2)
Collecting gast~=0.3.3
  Downloading gast-0.3.3-py2.py3-none-any.whl (9.7 kB)
Requirement already satisfied: astunparse~=1.6.3 in /usr/local/lib/python3.8/dist-packages (from tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (1.6.3)
Collecting pydantic!=1.8.1,!=1.8.2,<1.11.0,>=1.7.4
  Downloading pydantic-1.10.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (13.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.1/13.1 MB 52.7 MB/s eta 0:00:00
  Downloading pydantic-1.10.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (13.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.1/13.1 MB 57.1 MB/s eta 0:00:00
  Downloading pydantic-1.10.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (13.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.1/13.1 MB 54.4 MB/s eta 0:00:00
  Downloading pydantic-1.9.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 80.6 MB/s eta 0:00:00
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.8/dist-packages (from tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (2.16.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.8/dist-packages (from tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (0.6.1)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (1.0.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.8/dist-packages (from tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (0.4.6)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.8/dist-packages (from tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (3.4.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.8/dist-packages (from tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (1.8.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (2.10)
Requirement already satisfied: charset-normalizer<3,>=2 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (2.1.1)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (2022.9.24)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (1.24.3)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.8/dist-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (1.3.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from google-auth<3,>=1.6.3->tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (5.2.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.8/dist-packages (from google-auth<3,>=1.6.3->tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.8/dist-packages (from google-auth<3,>=1.6.3->tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (4.9)
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.8/dist-packages (from markdown>=2.6.8->tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (4.13.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.8/dist-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (3.10.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.8/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.8/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.4->tensorflow>2.1.0,>=2.0.0->adl101>0.6.0,>=0.1.5->machine-learning-datasets) (3.2.2)
```

Loading the Libraries

```
import math
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import warnings
warnings.filterwarnings('ignore')
import machine_learning_datasets as mldatasets
import numpy as np
from sklearn import preprocessing
import tensorflow as tf
tf.keras.backend.set_floatx('float32')
from tensorflow.keras.utils import get_file
from tensorflow.keras.models import load_model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout
import matplotlib.pyplot as plt
import seaborn as sns
from art.estimators.classification import TensorFlowV2Classifier
from art.estimators.certification.randomized_smoothing import TensorFlowV2RandomizedSmoothing
from art.utils import compute_accuracy
```

Understanding and Preparing the Data

```
X_train, X_test, y_train, y_test =\
                                mldatasets.load("maskedface-net_thumbs_sampled", prepare=True)
X_train, X_test = X_train / 255.0, X_test / 255.0
min_ = X_train.min()
max_ = X_train.max()
```

```
https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python/raw/master/Datasets/maskedface-net_thumbs_sampled.zip downloaded to /Users/smasis/Documents/OTHER/InterpretableMLBook/prog
/Users/smasis/Documents/OTHER/InterpretableMLBook/programming/Chapter13/data/maskedface-net_thumbs_sampled.zip uncompressed to /Users/smasis/Documents/OTHER/InterpretableMLBook/programming/Chapter13/d
21000 dataset files found in /Users/smasis/Documents/OTHER/InterpretableMLBook/programming/Chapter13/data/maskedface-net_thumbs_sampled folder
```

```
print('X_train dim:\t%s' % (X_train.shape,))
print('X_test dim:\t%s' % (X_test.shape,))
print('y_train dim:\t%s' % (y_train.shape,))
print('y_test dim:\t%s' % (y_test.shape,))
print('X_train min:\t%s' % (min_))
print('X_train max:\t%s' % (max_))
print('y_train labels:\t%s' % (np.unique(y_train)))
```

```
X_train dim:    (16800, 128, 128, 3)
X_test dim:     (4200, 128, 128, 3)
y_train dim:    (16800, 1)
y_test dim:     (4200, 1)
X_train min:    0.0
X_train max:    1.0
y_train labels: ['Correct' 'Incorrect' 'None']
```

```
ohe = preprocessing.OneHotEncoder(sparse=False)
ohe.fit(y_train)
labels_1 = ohe.categories_[0].tolist()
print(labels_1)
```

```
['Correct', 'Incorrect', 'None']
```

```
rand = 9
os.environ['PYTHONHASHSEED'] = str(rand)
tf.random.set_seed(rand)
np.random.seed(rand)
```

```
sampl_ind_idxs = np.random.choice(X_test.shape[0], 200, replace=False)
X_test_mdsample = X_test[sampl_ind_idxs].astype(np.float32)
y_test_mdsample = y_test[sampl_ind_idxs]
```

Evaluating and Certifying Adversarial Robustness

Certifying Robustness with Randomized Smoothing

```
[ ] nb_epochs = 10
    batch_size = 128
    optimizer = tf.keras.optimizers.Adam(lr=0.001)
    loss_object = tf.keras.losses.CategoricalCrossentropy()
    nb_classes = len(np.unique(y_train))
    sample_size = 100

    X_train, X_test_mdsample = X_train.astype(np.float32), X_test_mdsample.astype(np.float32)
    y_train_ohe = ohe.transform(y_train).astype(np.float32)
    y_test_mdsample_ohe = ohe.transform(y_test_mdsample).astype(np.float32)
```

```
    def get_model(input_shape, min_, max_):
        test_model = Sequential([
            Conv2D(16, (3, 3), activation='relu', input_shape=input_shape),
            MaxPooling2D(pool_size=(2, 2)),
            Conv2D(32, (3, 3), activation='relu'),
            MaxPooling2D(pool_size=(2, 2)),
            Conv2D(64, (3, 3), activation='relu'),
            MaxPooling2D(pool_size=(2, 2)),
            Conv2D(128, (5, 5), activation='relu'),
            MaxPooling2D(pool_size=(2, 2)),
            Flatten(),
            Dense(768, activation='relu'),
            Dropout(0.35),
            Dense(), activation='softmax')
        ])

        return test_model
```

```
[ ] def train_step(model, images, labels):
        with tf.GradientTape() as tape:
            predictions = model(images, training=True)
```

```
[ ] def train_step(model, images, labels):
        with tf.GradientTape() as tape:
            predictions = model(images, training=True)
            loss = loss_object(labels, predictions)
        gradients = tape.gradient(loss, model.trainable_variables)
        optimizer.apply_gradients(zip(gradients, model.trainable_variables))
```

```
    def train_rs_classifier(X_train, y_train, nb_epochs, batch_size, min_, max_,\
                            nb_classes, sample_size, loss_object, train_step, sigma=0, sigma_cert=0.5,\
                            alpha=0.001):

        input_shape = X_train.shape[1:]
        if sigma > 0:
            rs_classifier = TensorFlowV2RandomizedSmoothing(model=get_model(input_shape, min_, max_),\
                                                            input_shape=input_shape,\
                                                            clip_values=(min_, max_),\
                                                            nb_classes=nb_classes,\
                                                            sample_size=sample_size,\
                                                            loss_object=loss_object,\
                                                            train_step=train_step,\
                                                            scale=sigma, alpha=alpha,\
                                                            channels_first=False)

            rs_classifier.fit(X_train, y_train, nb_epochs=nb_epochs, batch_size=batch_size)

            return rs_classifier
        else:
            classifier = TensorFlowV2Classifier(model=get_model(input_shape, min_, max_),\
                                                input_shape=input_shape,\
                                                clip_values=(min_, max_),\
                                                nb_classes=nb_classes,\
                                                loss_object=loss_object,\
                                                train_step=train_step,\
                                                channels_first=False)

            classifier.fit(X_train, y_train, nb_epochs=nb_epochs, batch_size=batch_size)

            rs_classifier = TensorFlowV2RandomizedSmoothing(model=classifier.model,\
                                                            input_shape=input_shape,\
                                                            clip_values=(min_, max_),\
```

```
[ ]                                                     clip_values=(min_, max_),\
                                                        nb_classes=nb_classes,\
                                                        sample_size=sample_size,\
                                                        loss_object=loss_object,\
                                                        train_step=train_step,\
                                                        scale=sigma_cert, alpha=alpha,\
                                                        channels_first=False)
            return classifier, rs_classifier
```

```
sigma_0 = 0
classifier_0, rs_classifier_0 = train_rs_classifier(X_train, y_train_ohe, nb_epochs,\
                                                    batch_size, min_, max_, nb_classes,\
                                                    sample_size, loss_object, train_step,\
                                                    sigma_0)

sigma_1 = 0.25
rs_classifier_1 = train_rs_classifier(X_train, y_train_ohe, nb_epochs, batch_size,\
                                      min_, max_, nb_classes, sample_size, loss_object,\
                                      train_step, sigma_1)

sigma_2 = 0.5
rs_classifier_2 = train_rs_classifier(X_train, y_train_ohe, nb_epochs, batch_size,\
                                      min_, max_, nb_classes, sample_size, loss_object,\
                                      train_step, sigma_2)
```

```
CPU times: user 1h 36min 30s, sys: 4min 26s, total: 1h 40min 57s
Wall time: 12min 33s
```

```
[ ] y_preds_0 = classifier_0.predict(X_test_mdsample)
    y_preds_rs_1 = rs_classifier_1.predict(X_test_mdsample)
    y_preds_rs_2 = rs_classifier_2.predict(X_test_mdsample)
```

```
Randomized smoothing: 100%|██████████| 200/200 [01:03<00:00,  3.15it/s]
Randomized smoothing: 100%|██████████| 200/200 [01:03<00:00,  3.13it/s]
```

```
[ ] acc_0, cov_0 = compute_accuracy(y_preds_0, y_test_mdsample_ohe)
    acc_rs_1, cov_rs_1 = compute_accuracy(y_preds_rs_1, y_test_mdsample_ohe)
    acc_rs_2, cov_rs_2 = compute_accuracy(y_preds_rs_2, y_test_mdsample_ohe)

    print("Original Classifier")
```

```
[ ] print("Original Classifier")
    print("\tAccuracy: %.2f%%\tCoverage: %.2f%%" % (acc_0*100, cov_0*100))

    print("\nSmoothed Classifier (σ=%.2f)" % (sigma_1))
    print("\tAccuracy: %.2f%%\tCoverage: %.2f%%" % (acc_rs_1*100, cov_rs_1*100))

    print("\nSmoothed Classifier (σ=%.2f)" % (sigma_2))
    print("\tAccuracy: %.2f%%\tCoverage: %.2f%%" % (acc_rs_2*100, cov_rs_2*100))
```

```
Original Classifier
        Accuracy: 99.50%        Coverage: 100.00%

Smoothed Classifier (σ=0.25)
        Accuracy: 100.00%       Coverage: 99.50%

Smoothed Classifier (σ=0.50)
        Accuracy: 98.99%        Coverage: 99.50%
```

```
%%time
predictions_0, radiuses_0 = rs_classifier_0.certify(X_test_mdsample, n=500)
predictions_1, radiuses_1 = rs_classifier_1.certify(X_test_mdsample, n=500)
predictions_2, radiuses_2 = rs_classifier_2.certify(X_test_mdsample, n=500)
```

```
CPU times: user 1h 16min 3s, sys: 8min 31s, total: 1h 24min 35s
Wall time: 20min 52s
```

```
[ ] def calc_cert_accuracy(radius_list, predictions, radiuses, y_test):

        cert_accuracy = []
        nb_certs = len(radiuses)

        for r in radius_list:
            r_idx = np.where(radiuses >= r)[0]
            y_test_subset = y_test[r_idx]
            cert_accuracy_r = np.sum(predictions[r_idx] == np.argmax(y_test_subset, axis=1)) / nb_certs
            cert_accuracy.append(cert_accuracy_r)

        return cert_accuracy
```

```
plt.figure(figsize=(14,9))
plt.plot(radius_list, cert_accuracy_0, 'r-', label='original')
plt.plot(radius_list, cert_accuracy_1,\
         '-', color='green', label='smoothed, $\sigma$=' + str(sigma_1))
plt.plot(radius_list, cert_accuracy_2,\
         '-', color='blue', label='smoothed, $\sigma$=' + str(sigma_2))
plt.xlabel('Radius', fontsize=14)
plt.ylabel('Certified Accuracy', fontsize=14)
plt.legend()
plt.show()
```