DATA1030_FinalProject

December 5, 2021

[1]:
```
import numpy as np
import pandas as pd
import matplotlib as mpl
mpl.rcParams['figure.dpi'] = 250
import matplotlib
from matplotlib import pylab as plt
import joblib
```

[2]:
```
import os

cwd = os.getcwd()
dirct = os.path.abspath(os.path.join(cwd,os.pardir))
```

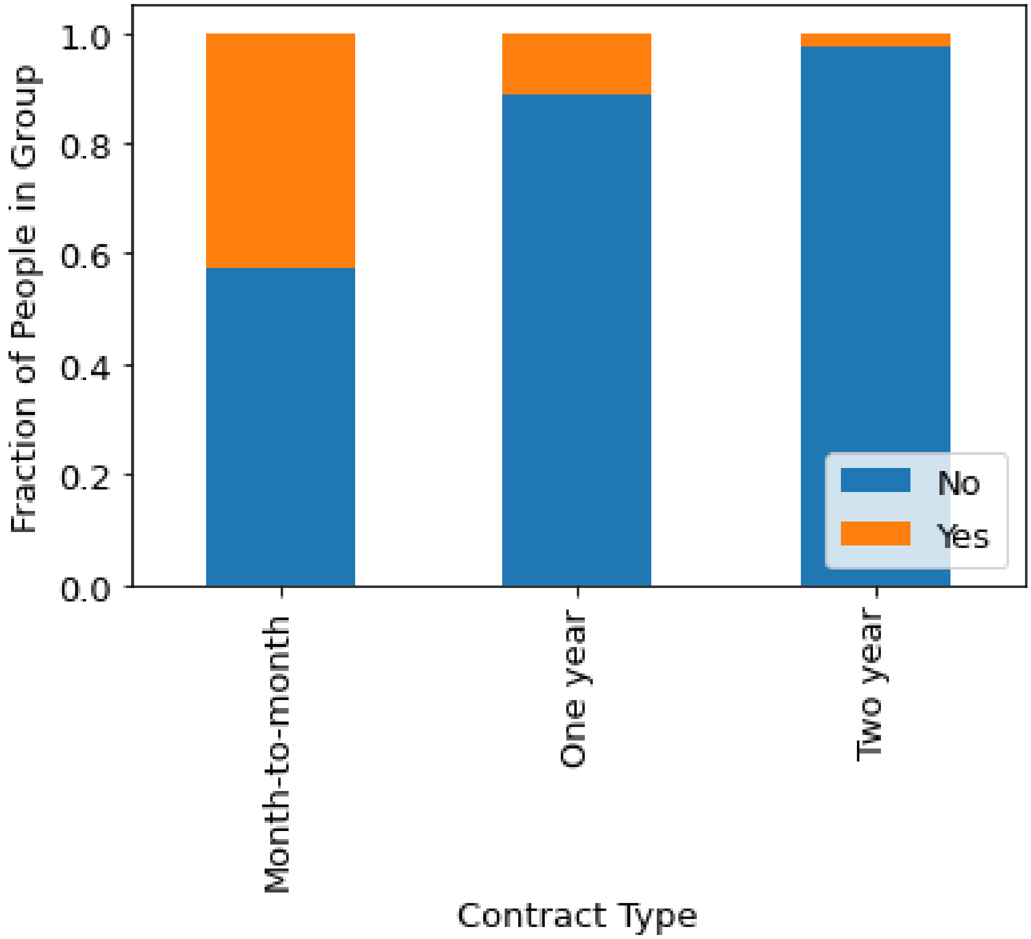
[919]:
```
df = pd.read_csv(dirct +'/data/WA_Fn-UseC_-Telco-Customer-Churn.csv')

# Exclude Customer ID
df = df.loc[:, df.columns != 'customerID']
```

[4]:
```
# number of rows
print(df.shape[0])

# number of columns
print(df.shape[1])
```

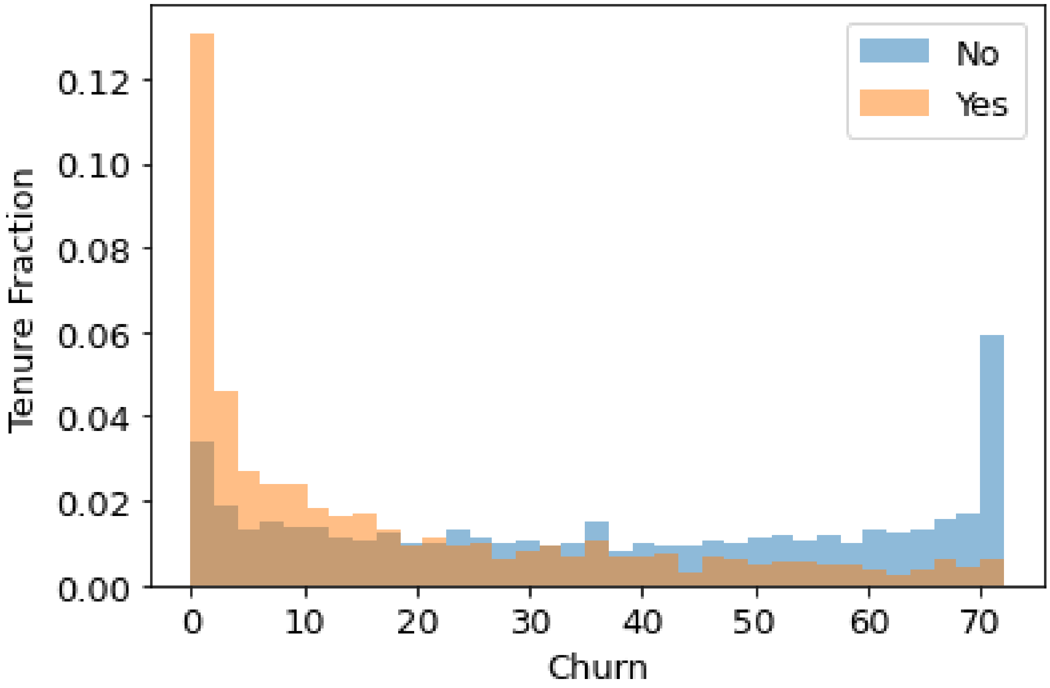
```
7043
20
```

1 1. Exploratory Data Analysis

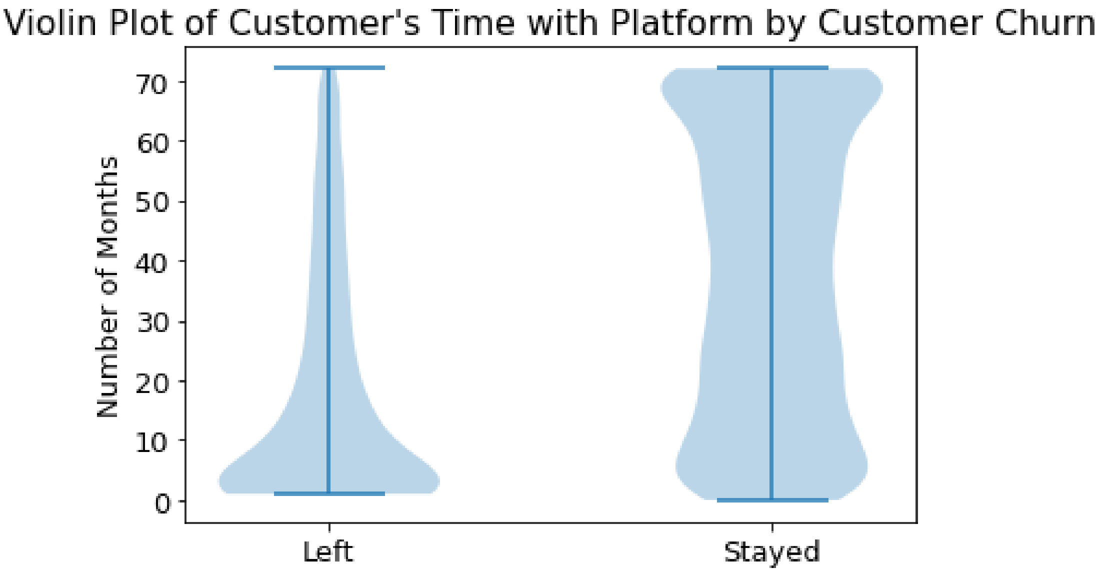
[921]:
```
# Churn by Contract Type



count_matrix = df.groupby(['Contract','Churn']).size().unstack()
count_matrix_norm = count_matrix.div(count_matrix.sum(axis=1),axis=0)
count_matrix_norm.plot(kind='bar', stacked=True)
plt.ylabel('Fraction of People in Group')
plt.xlabel('Contract Type')
plt.legend(loc=4)
plt.savefig(dirct +'/figures/ContractType_Churn.png',␣
 ↪bbox_inches='tight',dpi=300)
plt.show()
```



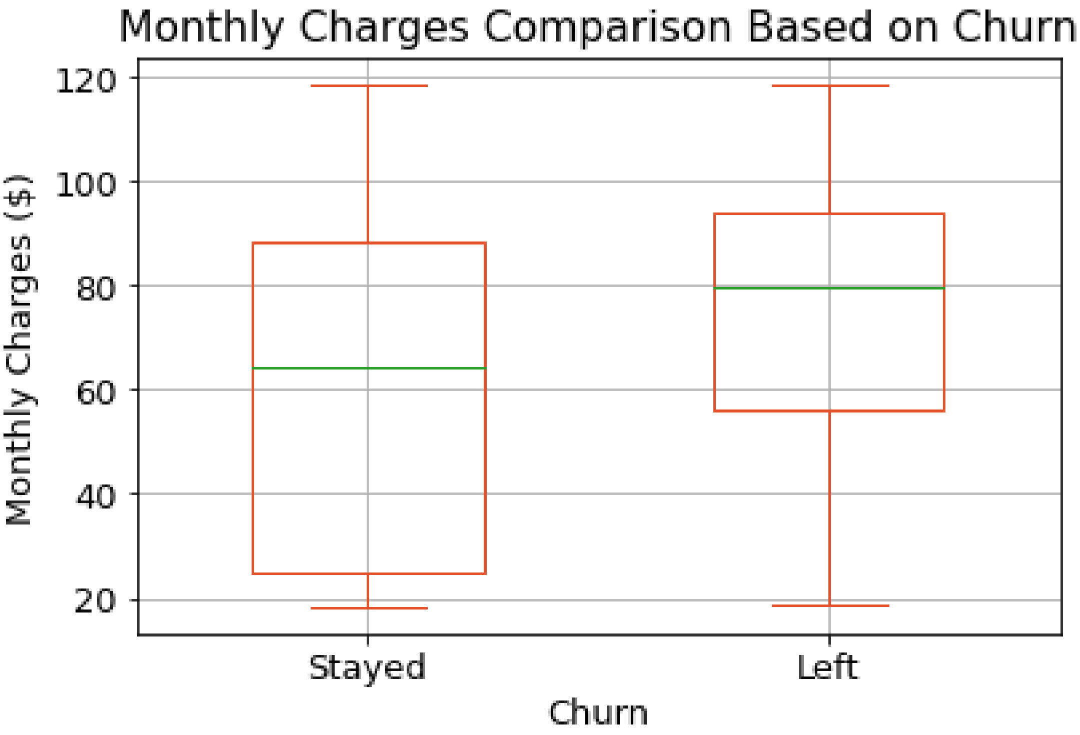
[924]:
```
df['tenure'].describe()

categories = df['Churn'].unique()
bin_range = (df['tenure'].min(),df['tenure'].max())

for c in categories:
    plt.hist(df[df['Churn']==c]['tenure'],alpha=0.
 ↪5,label=c,range=bin_range,bins=35,density=True)
plt.legend()
plt.ylabel('Tenure Fraction')
plt.xlabel('Churn')
plt.savefig(dirct +'/figures/tenure_Churn.png', bbox_inches='tight',dpi=300)
plt.show()
```

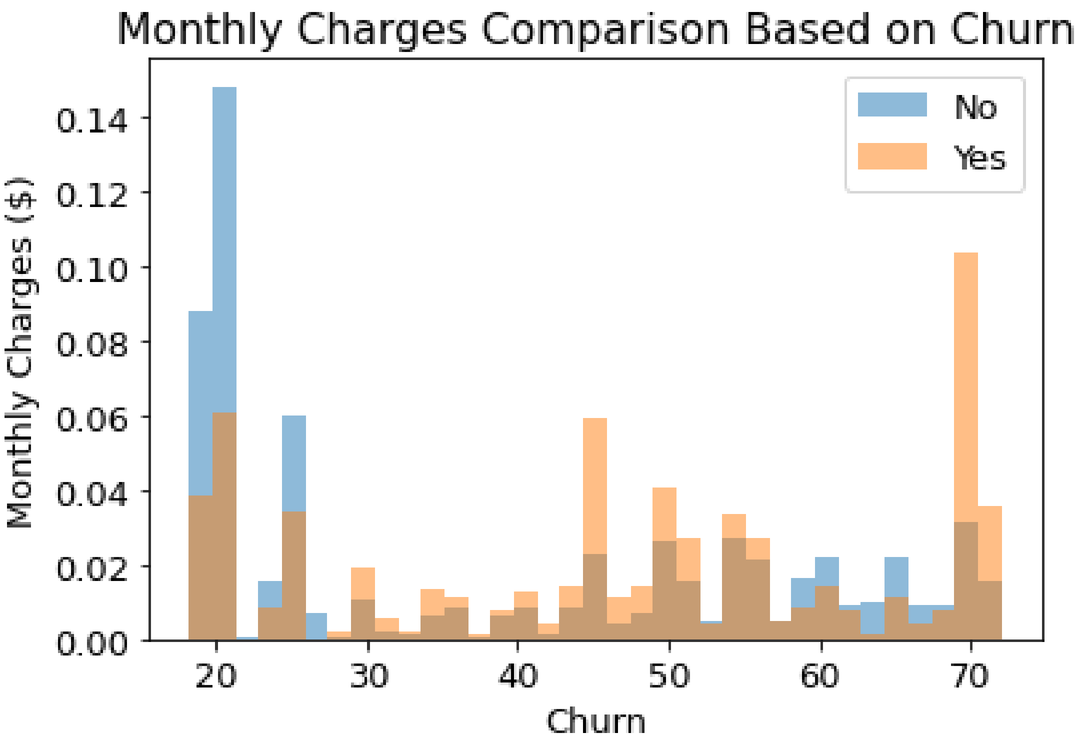


[925]:
```
dataset = [df[df['Churn']=='Yes']['tenure'].values,
           df[df['Churn']=='No']['tenure'].values]

plt.violinplot(dataset = dataset)
plt.xticks([1,2],['Left','Stayed'])
plt.ylabel('Number of Months')
plt.title("Violin Plot of Customer's Time with Platform by Customer Churn")
plt.savefig(dirct +'/figures/Violin_tenure_Churn.png',␣
 ↪bbox_inches='tight',dpi=300)
plt.show()
```

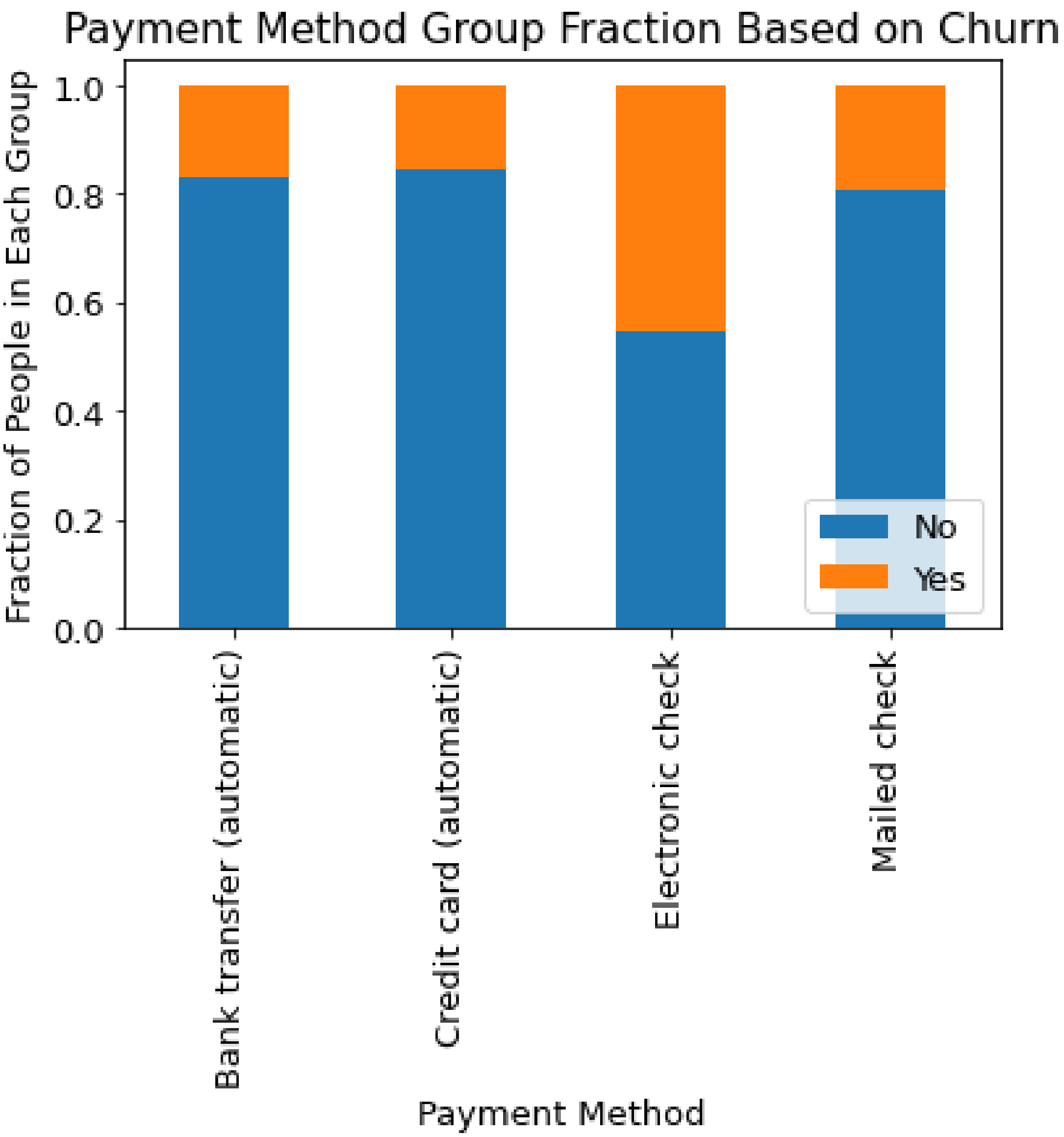


[929]:
```
colors = ['#4D3425','#E4512B']

df['MonthlyCharges'].describe()

df[['MonthlyCharges','Churn']].boxplot(by='Churn',widths=(0.5,0.5),
                                        boxprops=dict(color=colors[1]),␣
 ↪capprops=dict(color=colors[1]),whiskerprops=dict(color=colors[1]))
plt.ylabel('Monthly Charges ($)')
plt.xlabel('Churn')
plt.xticks([1,2],['Stayed','Left'])
plt.suptitle('')
plt.title('Monthly Charges Comparison Based on Churn')
plt.savefig(dirct +'/figures/boxplot_MonthlyCharges_Churn.png',␣
 ↪bbox_inches='tight',dpi=300)
plt.show()


categories = df['Churn'].unique()
bin_range = (df['MonthlyCharges'].min(),df['tenure'].max())

for c in categories:
    plt.hist(df[df['Churn']==c]['MonthlyCharges'],alpha=0.
 ↪5,label=c,range=bin_range,bins=35,density=True)
plt.legend()
plt.ylabel('Monthly Charges ($)')
plt.xlabel('Churn')
plt.suptitle('')
plt.title('Monthly Charges Comparison Based on Churn')
plt.savefig(dirct +'/figures/hstgm_MonthlyCharges_Churn.png',␣
 ↪bbox_inches='tight',dpi=300)
plt.show()
```



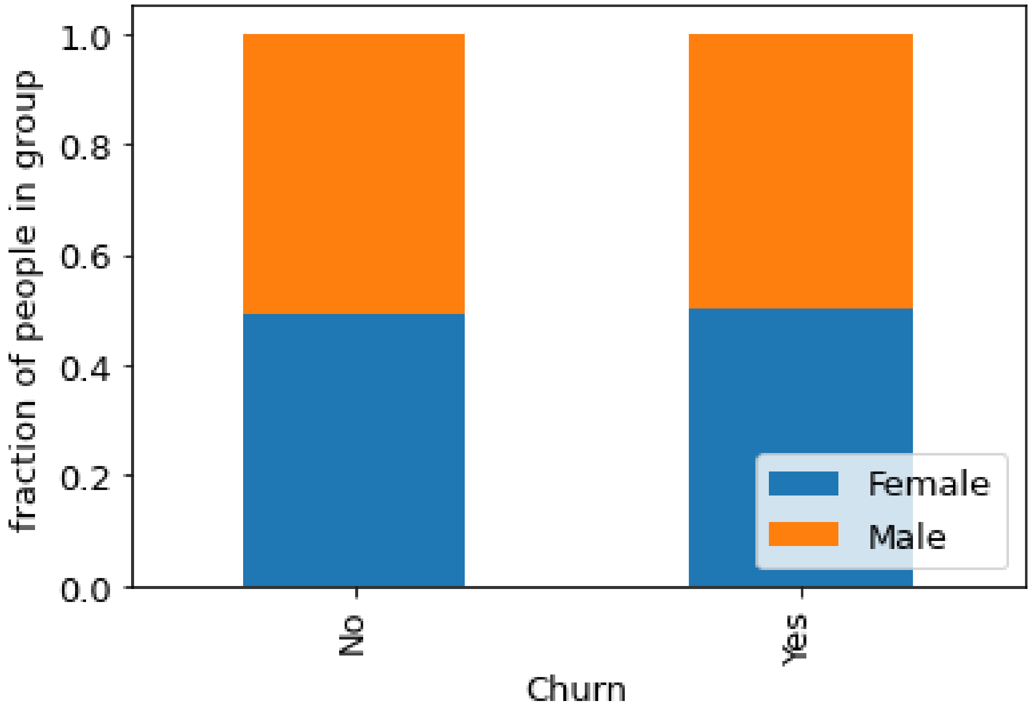


[930]:
```
# Churn by Payment Method

count_matrix = df.groupby(['PaymentMethod','Churn']).size().unstack()
count_matrix_norm = count_matrix.div(count_matrix.sum(axis=1),axis=0)
count_matrix_norm.plot(kind='bar', stacked=True)
plt.ylabel('Fraction of People in Each Group')
plt.xlabel('Payment Method')
plt.title('Payment Method Group Fraction Based on Churn')
plt.legend(loc=4)
plt.savefig(dirct +'/figures/paymentMethod_Churn.png',␣
 ↪bbox_inches='tight',dpi=300)
plt.show()
```

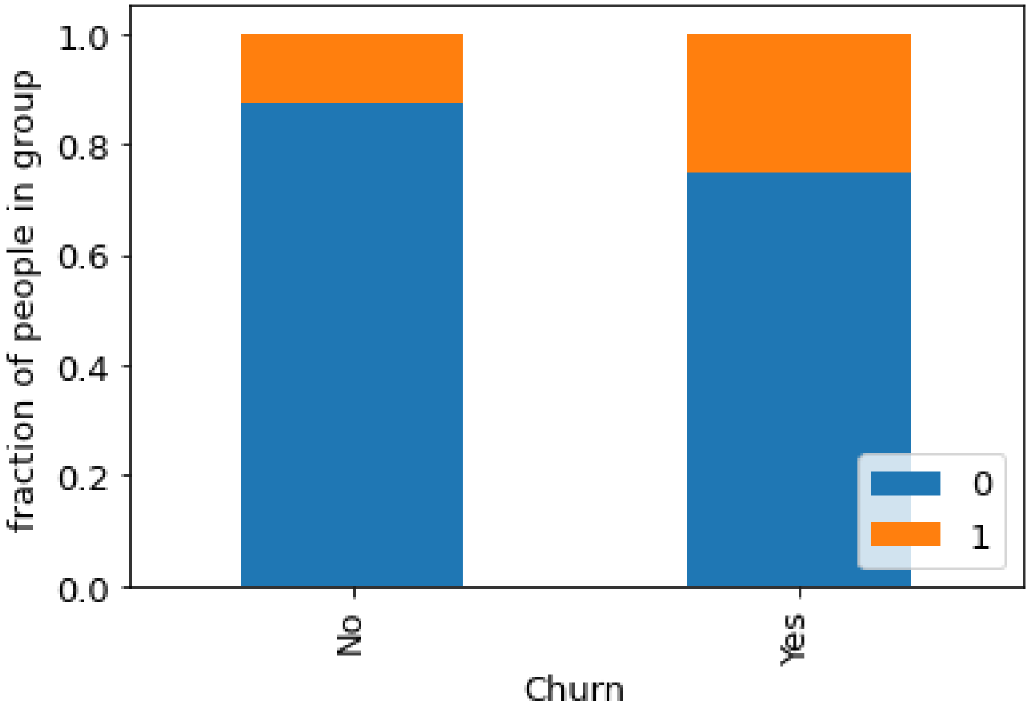


[920]:

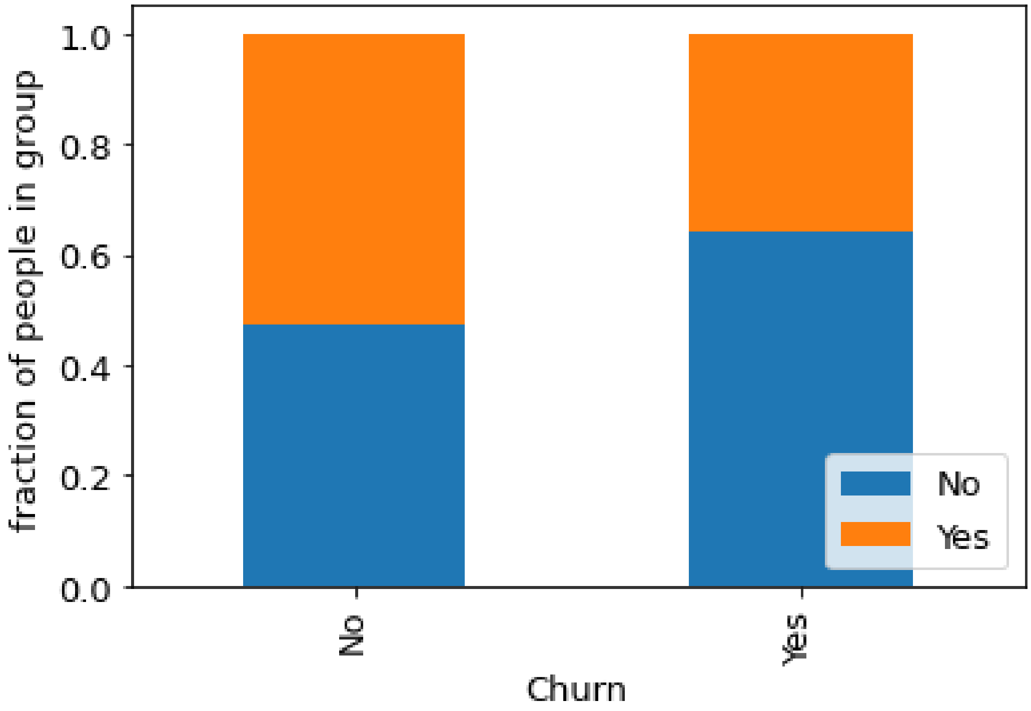
```
# If the feature is categorical, make bar graph, and print out values counts in␣
 ↪percentage and numbers
# If the feature is continuous/ numerical, make box plot, and print out␣
 ↪description

for col in df.columns:

    # categorical vs. continuous
    if df[col].dtypes == "float64":
        df[['Churn',col]].boxplot(by='Churn')
        plt.ylabel(col)
        plt.suptitle('')
        plt.title(col + ' Grouped by Churn Category')
        plt.show()
        print(df[col].describe())
        df[col].plot.hist(bins = int(np.sqrt(df.shape[0])))
        plt.xlabel(col)
        plt.ylabel('Count')
        plt.show()

    # categorical vs. categorical
    elif df[col].dtypes == "object":
        count_matrix = df.groupby(['Churn', col]).size().unstack()
        count_matrix_norm = count_matrix.div(count_matrix.sum(axis=1),axis=0)
        count_matrix_norm.plot(kind='bar', stacked=True)
        plt.ylabel('fraction of people in group')
        plt.legend(loc=4)
        plt.show()
        print(df[col].value_counts())
        print(df[col].value_counts(normalize=True))

    # categorical vs. cateogrical
    elif (df[col].dtypes == "int64") & (len(df[col].value_counts()) <= 15):
        count_matrix = df.groupby(['Churn', col]).size().unstack()
        count_matrix_norm = count_matrix.div(count_matrix.sum(axis=1),axis=0)
        count_matrix_norm.plot(kind='bar', stacked=True)
        plt.ylabel('fraction of people in group')
        plt.legend(loc=4)
        plt.show()
        print(df[col].value_counts())
        print(df[col].value_counts(normalize=True))

    # cateogrical vs. continuous
    elif (df[col].dtypes == "int64") & (len(df[col].value_counts()) > 15):
        df[['Churn',col]].boxplot(by='Churn')
        plt.ylabel(col)
        plt.suptitle('')
        plt.title(col + ' Grouped by Churn Category')
        plt.show()
        print(df[col].describe())
        df[col].plot.hist(bins = int(np.sqrt(df.shape[0])))
        plt.xlabel(col)
        plt.ylabel('Count')
        plt.show()
```



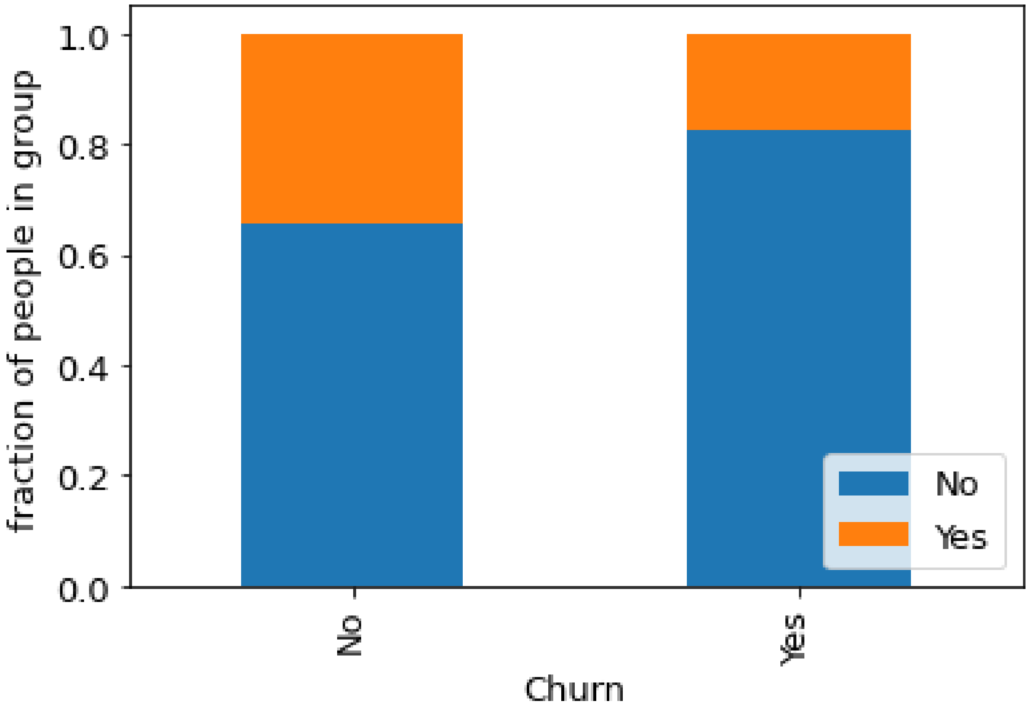
```
Male      3555
Female    3488
Name: gender, dtype: int64
Male      0.504756
Female    0.495244
Name: gender, dtype: float64
```



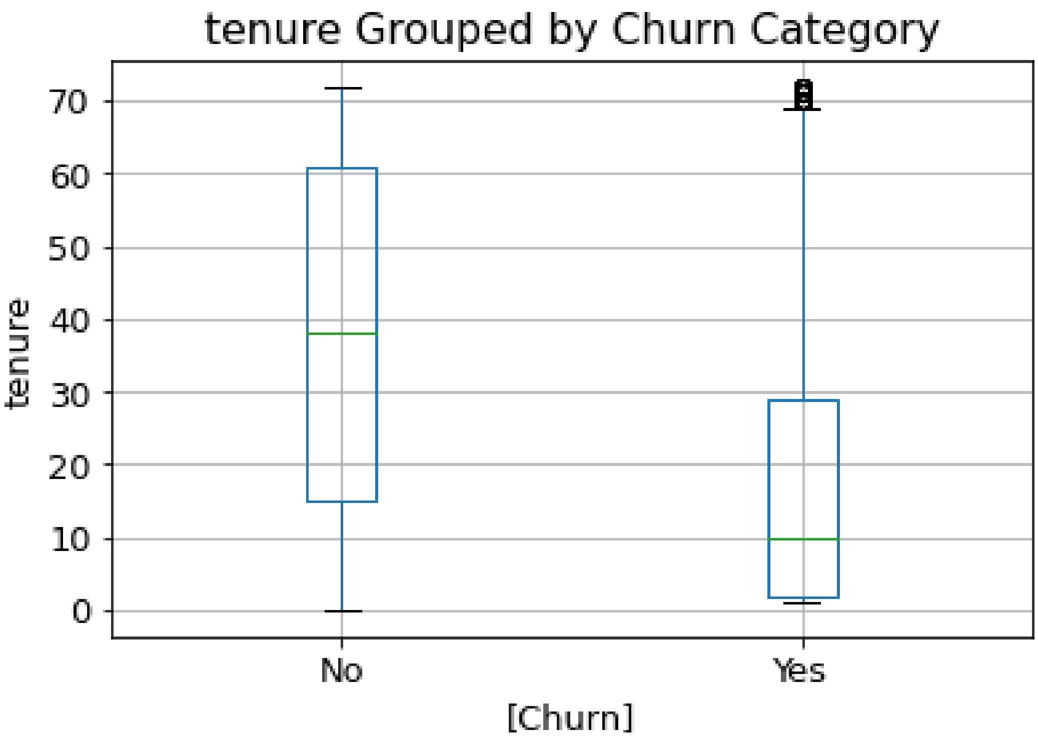
```
0    5901
1    1142
Name: SeniorCitizen, dtype: int64
0    0.837853
1    0.162147
Name: SeniorCitizen, dtype: float64
```



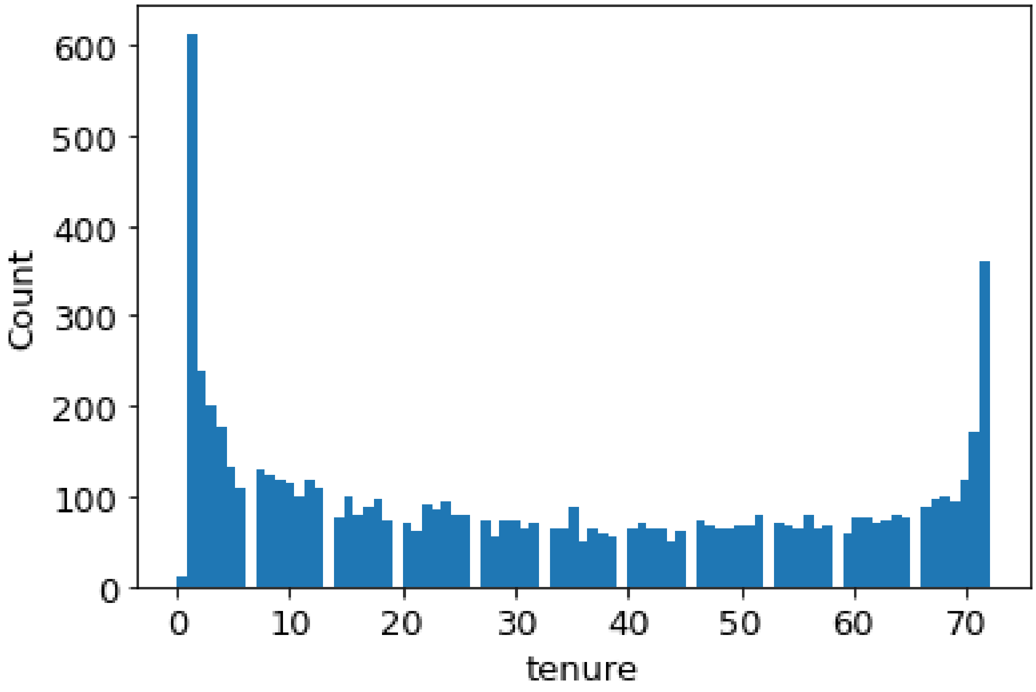
```
No     3641
Yes    3402
Name: Partner, dtype: int64
No     0.516967
Yes    0.483033
Name: Partner, dtype: float64
```

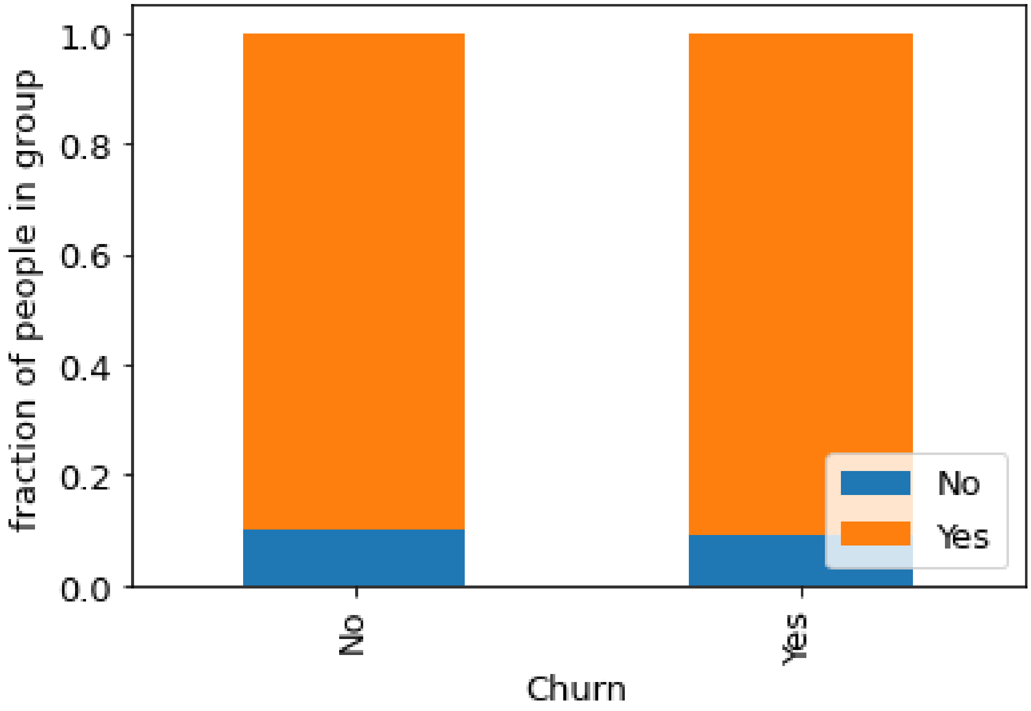


```
No     4933
Yes    2110
Name: Dependents, dtype: int64
No     0.700412
Yes    0.299588
Name: Dependents, dtype: float64
```



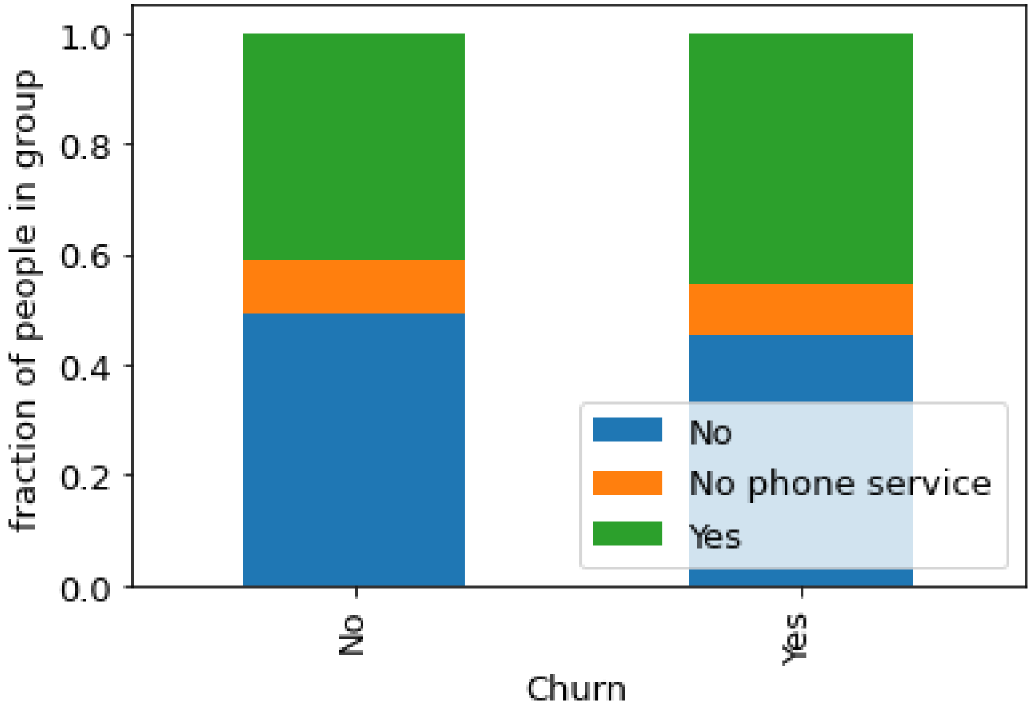
```
count    7043.000000
mean       32.371149
std        24.559481
min         0.000000
25%         9.000000
50%        29.000000
75%        55.000000
max        72.000000
Name: tenure, dtype: float64
```


Count
600
500
400
300
200
100
0
0
10
20
30
40
50
60
70
tenure

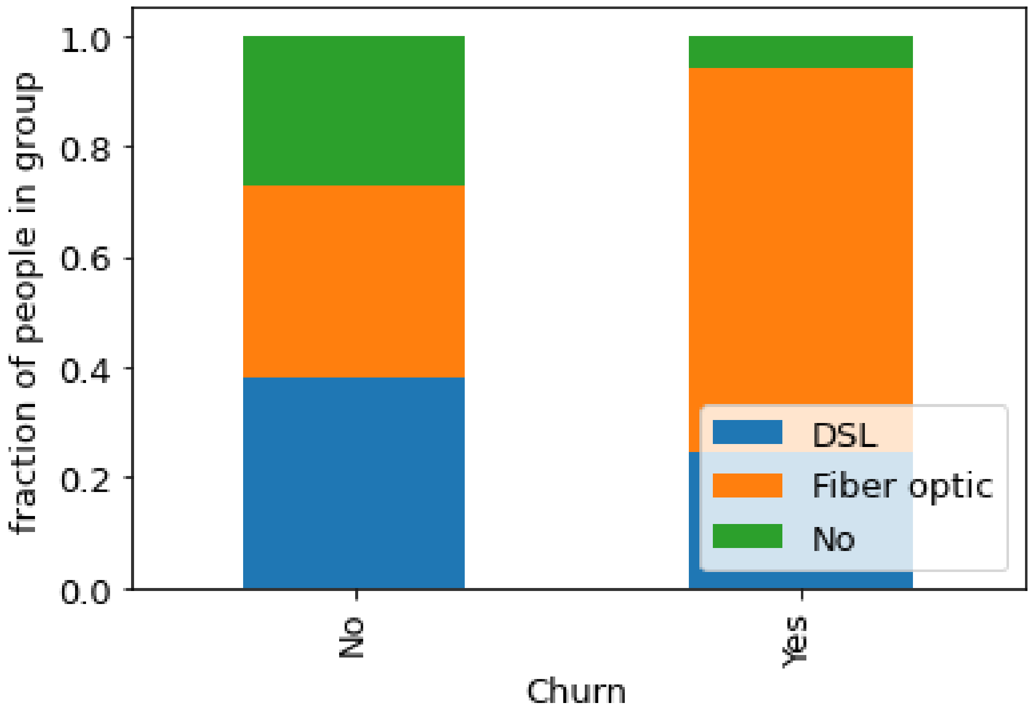

fraction of people in group
1.0
0.8
0.6
0.4
0.2
0.0
No
Yes
Churn
No
Yes

```
Yes     6361
```

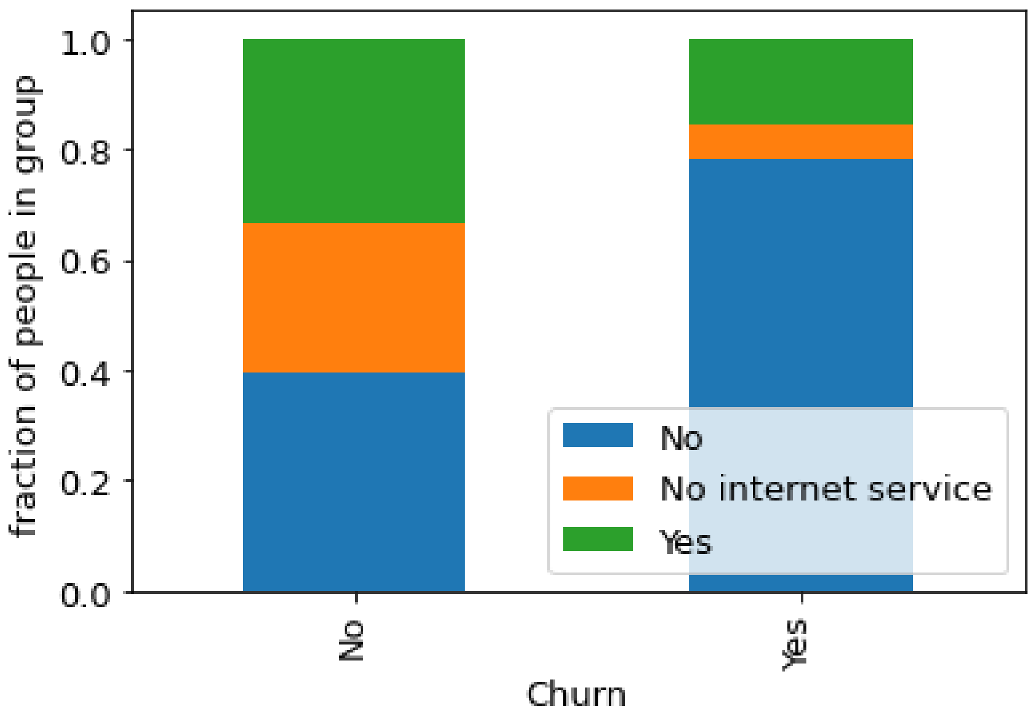
```
No      682
Name: PhoneService, dtype: int64
Yes    0.903166
No     0.096834
Name: PhoneService, dtype: float64
```



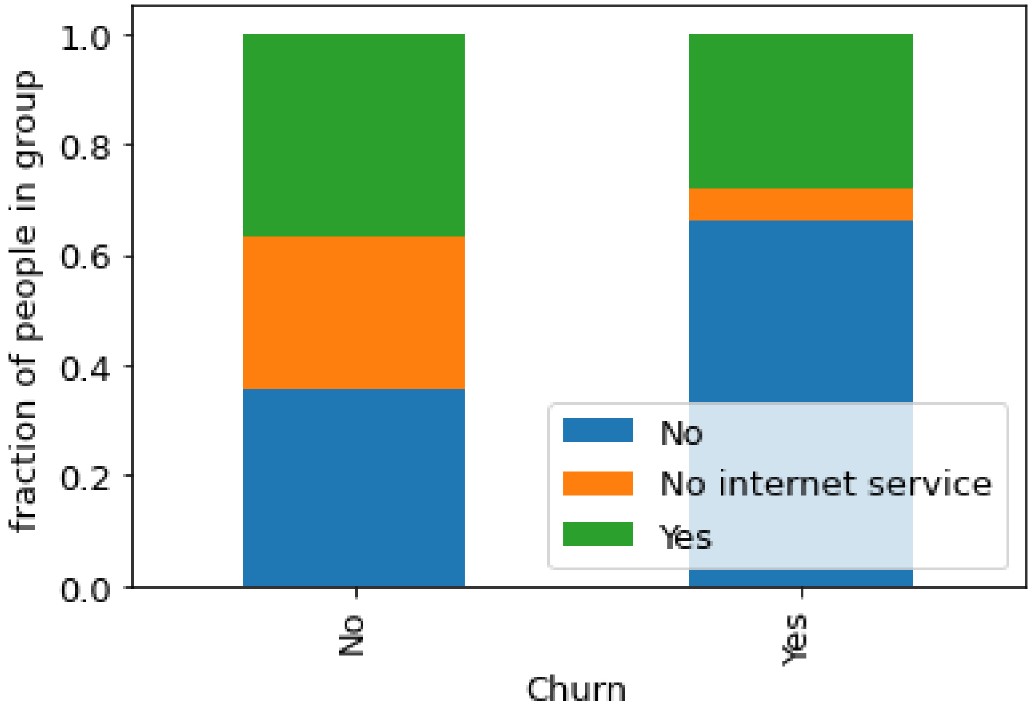
```
No                  3390
Yes                 2971
No phone service     682
Name: MultipleLines, dtype: int64
No                  0.481329
Yes                 0.421837
No phone service    0.096834
Name: MultipleLines, dtype: float64
```



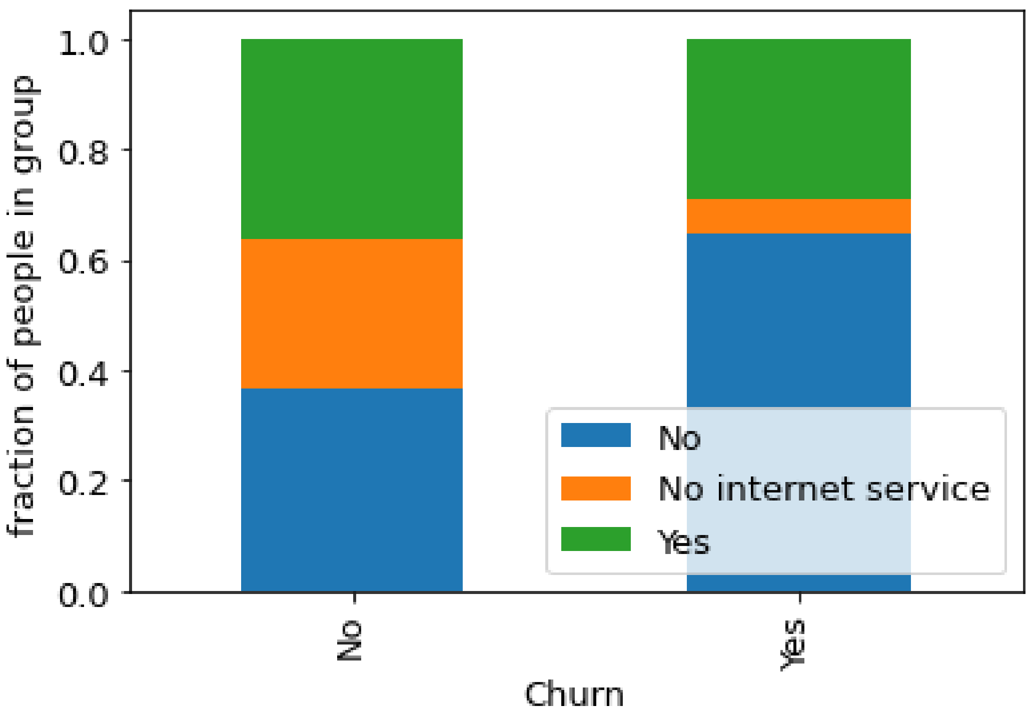
```
Fiber optic    3096
DSL            2421
No             1526
Name: InternetService, dtype: int64
Fiber optic    0.439585
DSL            0.343746
No             0.216669
Name: InternetService, dtype: float64
```



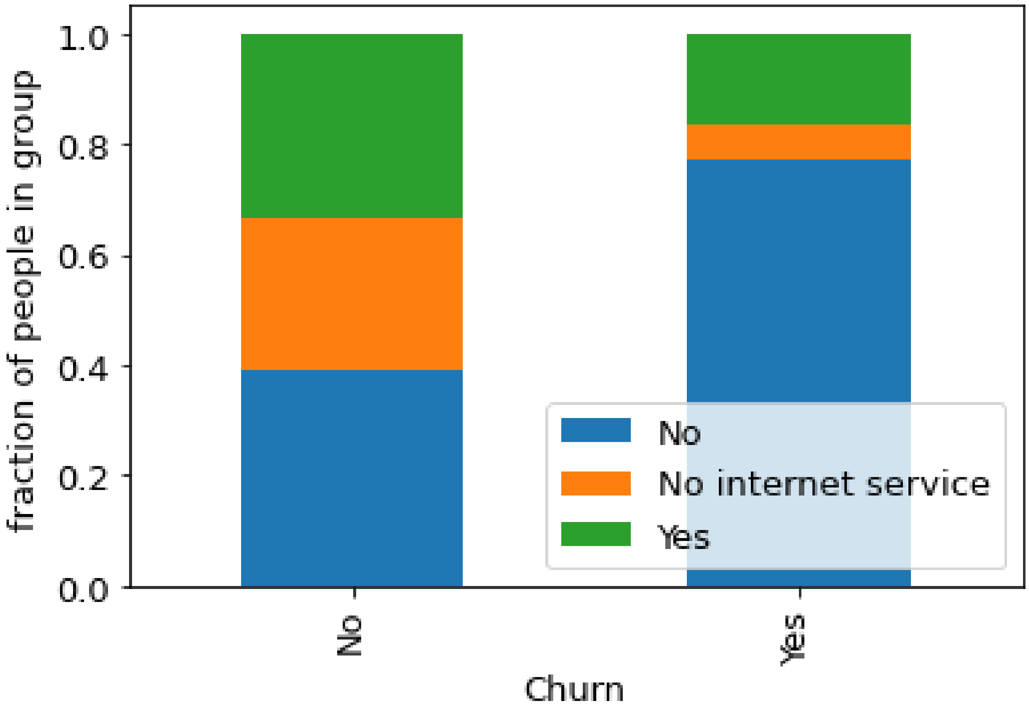
```
No                     3498
Yes                    2019
No internet service    1526
Name: OnlineSecurity, dtype: int64
No                     0.496663
Yes                    0.286668
No internet service    0.216669
Name: OnlineSecurity, dtype: float64
```



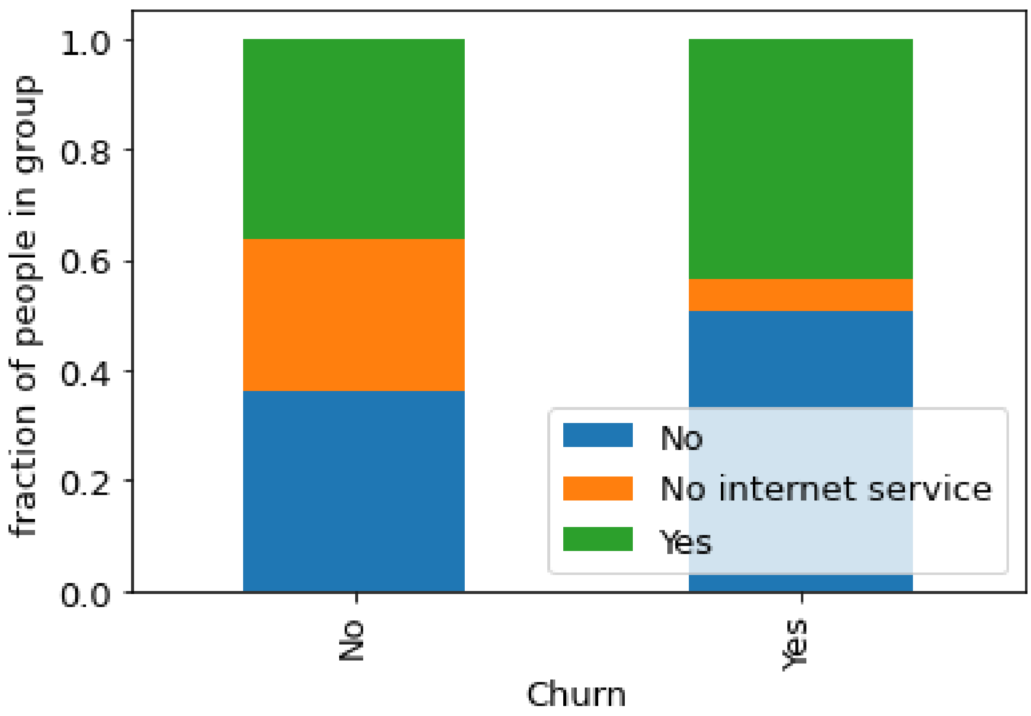
```
No                          3088
Yes                         2429
No internet service         1526
Name: OnlineBackup, dtype: int64
No                          0.438450
Yes                         0.344881
No internet service         0.216669
Name: OnlineBackup, dtype: float64
```



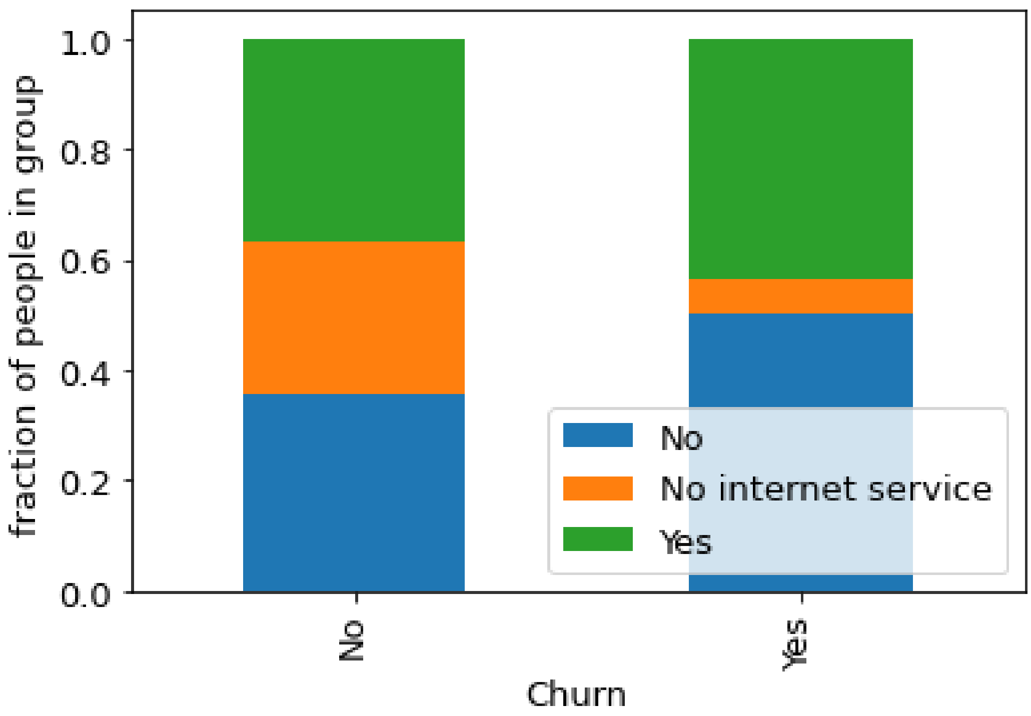
```
No                     3095
Yes                    2422
No internet service    1526
Name: DeviceProtection, dtype: int64
No                     0.439443
Yes                    0.343888
No internet service    0.216669
Name: DeviceProtection, dtype: float64
```



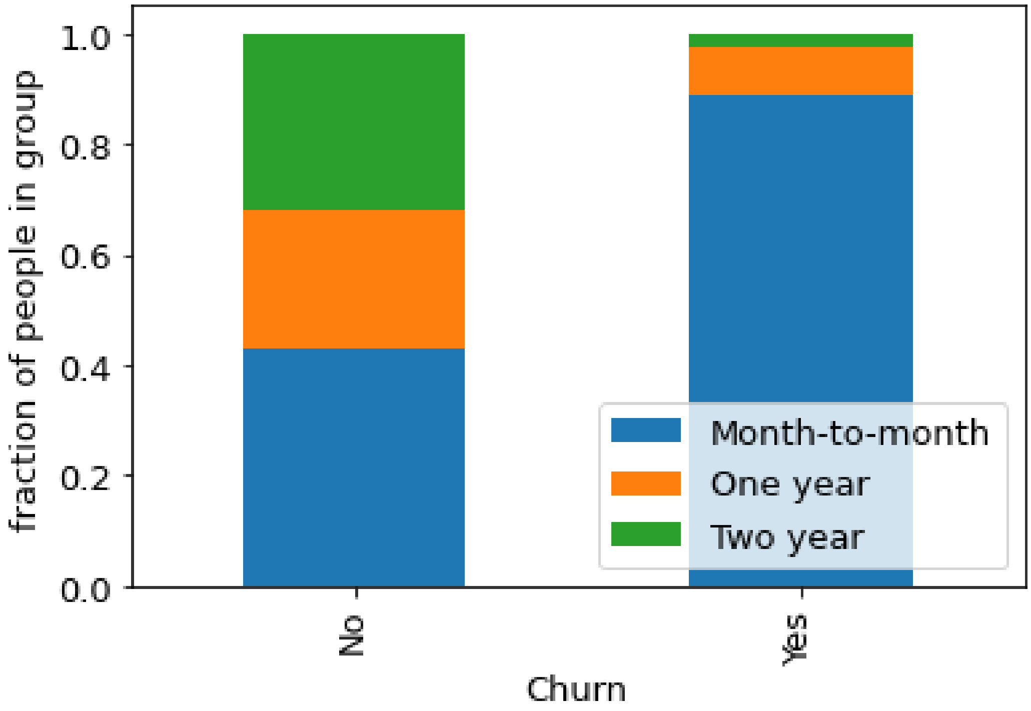
```
No                     3473
Yes                    2044
No internet service    1526
Name: TechSupport, dtype: int64
No                     0.493114
Yes                    0.290217
No internet service    0.216669
Name: TechSupport, dtype: float64
```



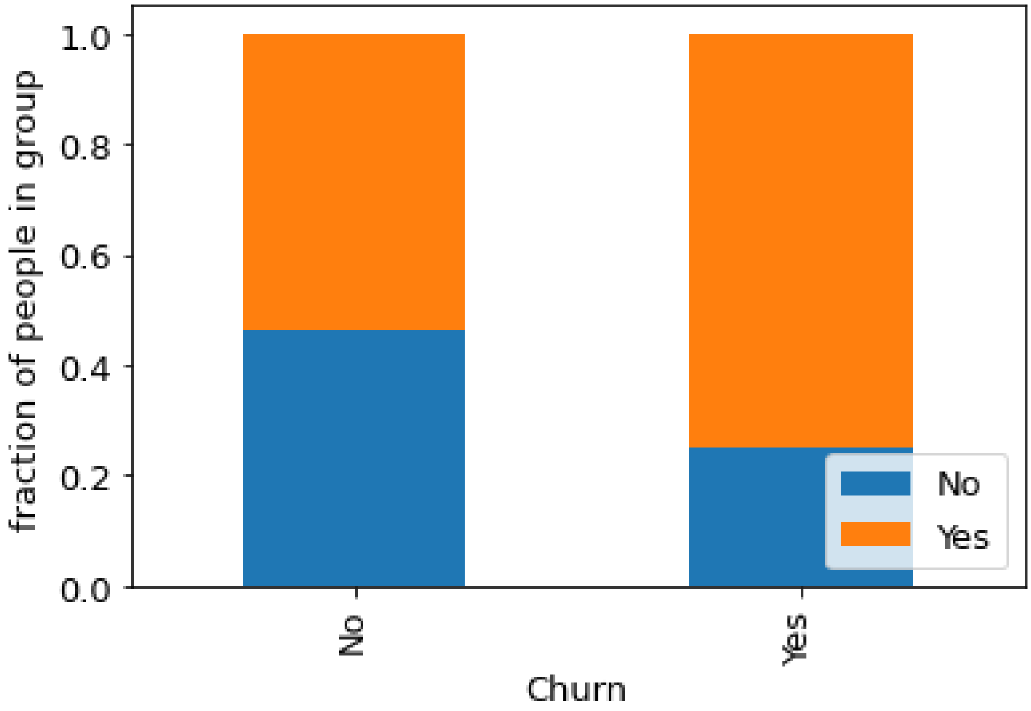
```
No                     2810
Yes                    2707
No internet service    1526
Name: StreamingTV, dtype: int64
No                     0.398978
Yes                    0.384353
No internet service    0.216669
Name: StreamingTV, dtype: float64
```



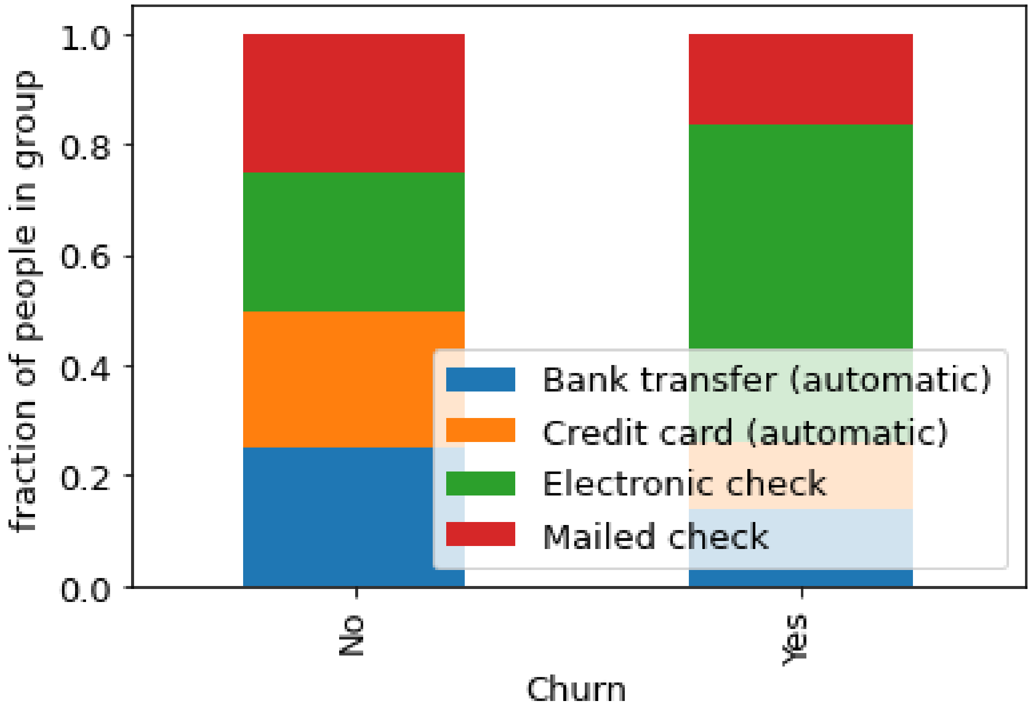
```
No                         2785
Yes                        2732
No internet service        1526
Name: StreamingMovies, dtype: int64
No                         0.395428
Yes                        0.387903
No internet service        0.216669
Name: StreamingMovies, dtype: float64
```



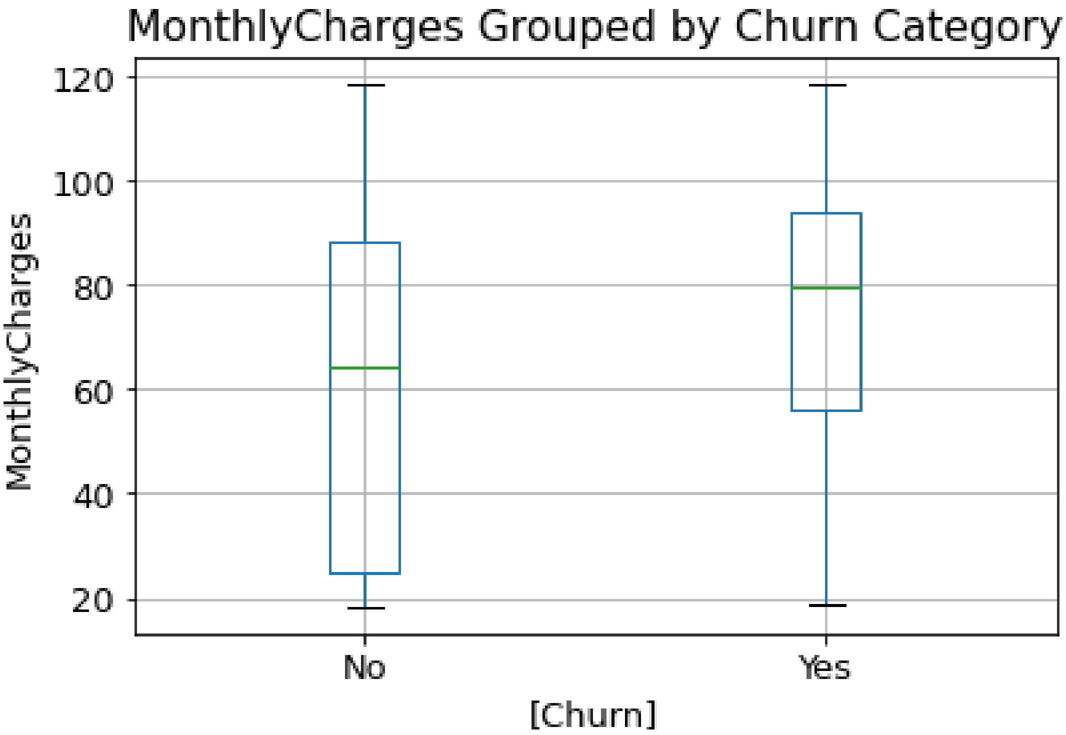
```
Month-to-month    3875
Two year          1695
One year          1473
Name: Contract, dtype: int64
Month-to-month    0.550192
Two year          0.240664
One year          0.209144
Name: Contract, dtype: float64
```



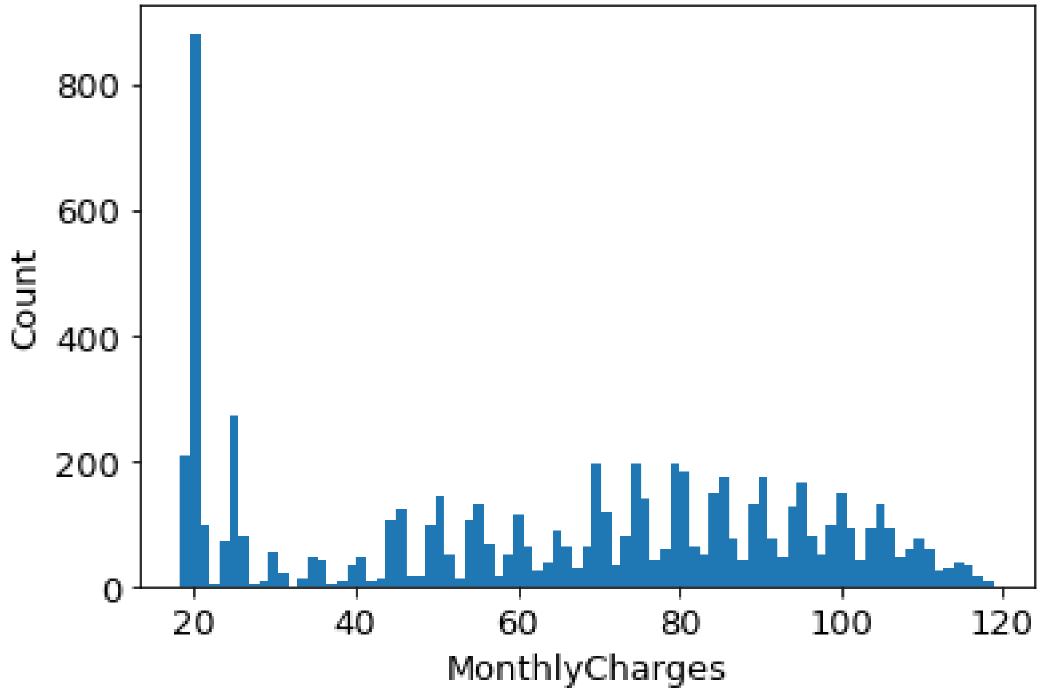
```
Yes    4171
No     2872
Name: PaperlessBilling, dtype: int64
Yes    0.592219
No     0.407781
Name: PaperlessBilling, dtype: float64
```



```
Electronic check             2365
Mailed check                 1612
Bank transfer (automatic)    1544
Credit card (automatic)      1522
Name: PaymentMethod, dtype: int64
Electronic check             0.335794
Mailed check                 0.228880
Bank transfer (automatic)    0.219225
Credit card (automatic)      0.216101
Name: PaymentMethod, dtype: float64
```



```
count    7043.000000
mean       64.761692
std        30.090047
min        18.250000
25%        35.500000
50%        70.350000
75%        89.850000
max       118.750000
Name: MonthlyCharges, dtype: float64
```



```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/var/folders/r0/t3r63gk55yv_v1n43ch_4shc0000gp/T/ipykernel_5261/846890756.py in
 ↪<module>
     24         plt.ylabel('fraction of people in group')
     25         plt.legend(loc=4)
---> 26         plt.show()
     27         print(df[col].value_counts())
     28         print(df[col].value_counts(normalize=True))

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/pyplot.py i
 ↪show(*args, **kwargs)
    376     """
    377     _warn_if_gui_out_of_main_thread()
--> 378     return _backend_mod.show(*args, **kwargs)
    379
    380

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib_inline/
 ↪backend_inline.py in show(close, block)
     39     try:
     40         for figure_manager in Gcf.get_all_fig_managers():
---> 41             display(

     42                 figure_manager.canvas.figure,
     43                 metadata=_fetch_figure_metadata(figure_manager.canvas.
 ↪figure)

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/IPython/core/display.p␣
 ↪in display(include, exclude, metadata, transient, display_id, *objs, **kwargs
    318             publish_display_data(data=obj, metadata=metadata, **kwargs)
    319         else:
--> 320             format_dict, md_dict = format(obj, include=include,␣
 ↪exclude=exclude)
    321             if not format_dict:
    322                 # nothing to display (e.g. _ipython_display_ took over)

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/IPython/core/formatters.
 ↪py in format(self, obj, include, exclude)
    178             md = None
    179             try:
--> 180                 data = formatter(obj)
    181             except:
    182                 # FIXME: log the exception

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/decorator.py in␣
 ↪fun(*args, **kw)
    230             if not kwsyntax:
    231                 args, kw = fix(args, kw, sig)
--> 232             return caller(func, *(extras + args), **kw)
    233     fun.__name__ = func.__name__
    234     fun.__doc__ = func.__doc__

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/IPython/core/formatters.
 ↪py in catch_format_error(method, self, *args, **kwargs)
    222     """show traceback on failed format call"""
    223     try:
--> 224         r = method(self, *args, **kwargs)
    225     except NotImplementedError:
    226         # don't warn on NotImplementedErrors

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/IPython/core/formatters.
 ↪py in __call__(self, obj)
    339                 pass
    340             else:
--> 341                 return printer(obj)
    342             # Finally look for special method names
    343             method = get_real_method(obj, self.print_method)

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/IPython/core/pylabtools.
 ↪py in <lambda>(fig)
    251
    252     if 'png' in formats:
```

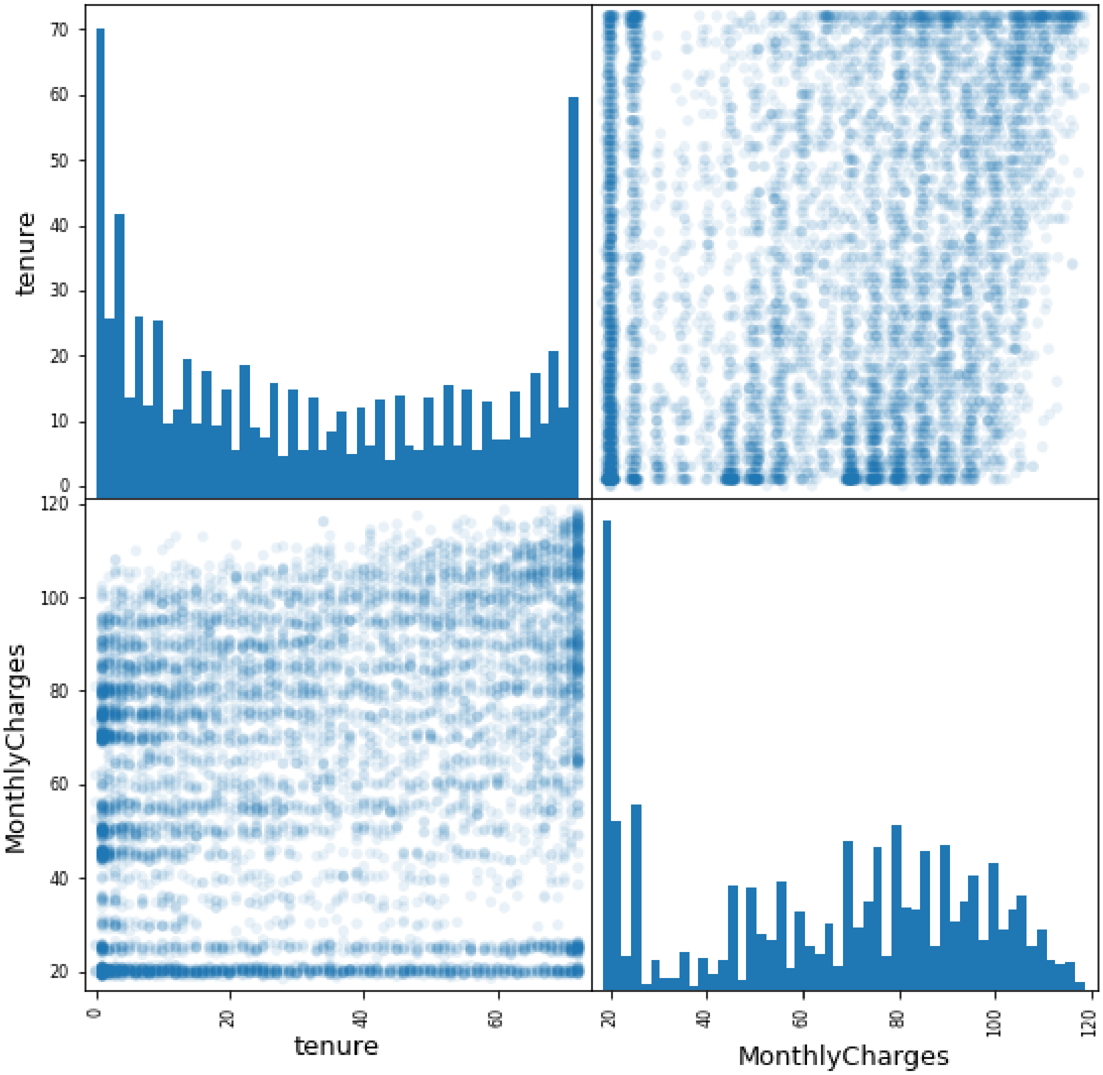
```
--> 253         png_formatter.for_type(Figure, lambda fig: print_figure(fig,␣
 ↪'png', **kwargs))
    254     if 'retina' in formats or 'png2x' in formats:
    255         png_formatter.for_type(Figure, lambda fig: retina_figure(fig,␣
 ↪**kwargs))

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/IPython/core/pylabtools.
 ↪py in print_figure(fig, fmt, bbox_inches, **kwargs)
    135         FigureCanvasBase(fig)
    136
--> 137     fig.canvas.print_figure(bytes_io, **kw)
    138     data = bytes_io.getvalue()
    139     if fmt == 'svg':

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/
 ↪backend_bases.py in print_figure(self, filename, dpi, facecolor, edgecolor,␣
 ↪orientation, format, bbox_inches, pad_inches, bbox_extra_artists, backend,␣
 ↪**kwargs)
   2228                     else suppress())
   2229                 with ctx:
-> 2230                     self.figure.draw(renderer)
   2231
   2232             if bbox_inches:

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/artist.py in␣
 ↪draw_wrapper(artist, renderer, *args, **kwargs)
     72     @wraps(draw)
     73     def draw_wrapper(artist, renderer, *args, **kwargs):
---> 74         result = draw(artist, renderer, *args, **kwargs)
     75         if renderer._rasterizing:
     76             renderer.stop_rasterizing()

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/artist.py in␣
 ↪draw_wrapper(artist, renderer, *args, **kwargs)
     49                 renderer.start_filter()
     50
---> 51             return draw(artist, renderer, *args, **kwargs)
     52         finally:
     53             if artist.get_agg_filter() is not None:

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/figure.py in␣
 ↪draw(self, renderer)
   2778
   2779             self.patch.draw(renderer)
-> 2780             mimage._draw_list_compositing_images(
   2781                 renderer, self, artists, self.suppressComposite)
   2782
```

```
/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/image.py in
 ↪_draw_list_compositing_images(renderer, parent, artists, suppress_composite)
    130     if not_composite or not has_images:
    131         for a in artists:
--> 132             a.draw(renderer)
    133     else:
    134         # Composite any adjacent images together

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/artist.py i
 ↪draw_wrapper(artist, renderer, *args, **kwargs)
     49                 renderer.start_filter()
     50 
---> 51             return draw(artist, renderer, *args, **kwargs)
     52         finally:
     53             if artist.get_agg_filter() is not None:

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/_api/
 ↪deprecation.py in wrapper(*inner_args, **inner_kwargs)
    429                          else deprecation_addendum,
    430                 **kwargs)
--> 431         return func(*inner_args, **inner_kwargs)
    432 
    433     return wrapper

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/axes/_base.
 ↪py in draw(self, renderer, inframe)
   2919             renderer.stop_rasterizing()
   2920 
-> 2921         mimage._draw_list_compositing_images(renderer, self, artists)
   2922 
   2923         renderer.close_group('axes')

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/image.py in
 ↪_draw_list_compositing_images(renderer, parent, artists, suppress_composite)
    130     if not_composite or not has_images:
    131         for a in artists:
--> 132             a.draw(renderer)
    133     else:
    134         # Composite any adjacent images together

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/artist.py i
 ↪draw_wrapper(artist, renderer, *args, **kwargs)
     49                 renderer.start_filter()
     50 
---> 51             return draw(artist, renderer, *args, **kwargs)
     52         finally:
     53             if artist.get_agg_filter() is not None:
```

```
/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/legend.py i
 ↪draw(self, renderer)
    612
    613         self.legendPatch.draw(renderer)
--> 614         self._legend_box.draw(renderer)
    615
    616         renderer.close_group('legend')

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/offsetbox.p
 ↪in draw(self, renderer)
    366         for c, (ox, oy) in zip(self.get_visible_children(), offsets):
    367             c.set_offset((px + ox, py + oy))
--> 368             c.draw(renderer)
    369
    370         bbox_artist(self, renderer, fill=False, props=dict(pad=0.))

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/offsetbox.p
 ↪in draw(self, renderer)
    366         for c, (ox, oy) in zip(self.get_visible_children(), offsets):
    367             c.set_offset((px + ox, py + oy))
--> 368             c.draw(renderer)
    369
    370         bbox_artist(self, renderer, fill=False, props=dict(pad=0.))

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/offsetbox.p
 ↪in draw(self, renderer)
    366         for c, (ox, oy) in zip(self.get_visible_children(), offsets):
    367             c.set_offset((px + ox, py + oy))
--> 368             c.draw(renderer)
    369
    370         bbox_artist(self, renderer, fill=False, props=dict(pad=0.))

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/offsetbox.p
 ↪in draw(self, renderer)
    359         to the given *renderer*.
    360         """
--> 361         width, height, xdescent, ydescent, offsets = self.
 ↪get_extent_offsets(
    362                                                         renderer)
    363

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/offsetbox.p
 ↪in get_extent_offsets(self, renderer)
    472         sep = self.sep * dpicor
    473
--> 474         whd_list = [c.get_extent(renderer)
    475                     for c in self.get_visible_children()]
    476

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/offsetbox.p␣
 ↪in <listcomp>(.0)
    472         sep = self.sep * dpicor
    473
--> 474         whd_list = [c.get_extent(renderer)
    475                     for c in self.get_visible_children()]
    476

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/offsetbox.p␣
 ↪in get_extent(self, renderer)
    821             ismath="TeX" if self._text.get_usetex() else False)
    822
--> 823         bbox, info, yd = self._text._get_layout(renderer)
    824         w, h = bbox.width, bbox.height
    825

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/text.py in␣
 ↪_get_layout(self, renderer)
    312             clean_line, ismath = self._preprocess_math(line)
    313             if clean_line:
--> 314                 w, h, d = renderer.get_text_width_height_descent(
    315                     clean_line, self._fontproperties, ismath=ismath)
    316             else:

/opt/anaconda3/envs/data1030/lib/python3.9/site-packages/matplotlib/backends/
 ↪backend_agg.py in get_text_width_height_descent(self, s, prop, ismath)
    238         flags = get_hinting_flag()
    239         font = self._get_agg_font(prop)
--> 240         font.set_text(s, 0.0, flags=flags)
    241         w, h = font.get_width_height()  # width and height of unrotated␣
 ↪string
    242         d = font.get_descent()

KeyboardInterrupt:
```

[931]:

```
# Only three continuous variables for scatter matrix

df_continuous = df[['tenure','MonthlyCharges','TotalCharges']]

pd.plotting.scatter_matrix(df_continuous, figsize=(9, 9),␣
 ↪marker='o',hist_kwds={'bins': 50},
                           s=30, alpha=.1)
plt.show()
```



[5]:

```
# Missing Values

## Change the type of Feature TotalCharges
df.TotalCharges = pd.to_numeric(df.TotalCharges, errors='coerce')
print(df['TotalCharges'].describe())

print(df.isnull().sum())
```

```
count    7032.000000
mean     2283.300441
std      2266.771362
min        18.800000
25%       401.450000
50%      1397.475000
```

```
75%      3794.737500
max      8684.800000
Name: TotalCharges, dtype: float64
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
Churn                0
dtype: int64
```

2 2. Methods

[6]:

```
# Missing Data: delete 7 rows of missing total charges variable

df_r = df.dropna()
print(df_r.isnull().sum())
```

```
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
```

```
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

[7]:
```
print(df_r.shape[0])
```

```
7032
```

[77]:
```
y = df_r['Churn']
X = df_r.loc[:, df_r.columns != 'Churn']
```

[160]:
```
# Calculate Baseline f1 Score
# Predict all classes to be 1

class_0 = df_r['Churn'].value_counts()[0]
class_1 = df_r['Churn'].value_counts()[1]

TN = 0
FP = class_0

print(class_0)

FN = 0
TP = class_1

print(class_1)

p = TP/(TP+FP)
r = TP/(TP+FN)

baseline_f1 = (2*p*r)/(p+r)

print("Hand-calculated baseline F1 score: ",baseline_f1)
```

```
5163
1869
Hand-calculated baseline F1 score:  0.41995281429052916
```

[159]:
```
y = df_r['Churn'].map(dict(Yes=1, No=0))
y_baseline =  np.full((7032, 1), 1)

from sklearn.metrics import f1_score
f1_score(y, y_baseline)

print("sklearn calculated baseline F1 score: ",baseline_f1)
```

```
sklearn calculated baseline F1 score:  0.41995281429052916
```

[78]:

```
y = df_r['Churn'].map(dict(Yes=1, No=0))
y
```

[78]:

```
0       0
1       0
2       1
3       0
4       1
       ..
7038    0
7039    0
7040    0
7041    1
7042    0
Name: Churn, Length: 7032, dtype: int64
```

[156]:

```
y = df_r['Churn'].map(dict(Yes=1, No=0))
y_baseline =  np.full((7032, 1), 1)

from sklearn.metrics import fbeta_score
fbeta_score(y, y_baseline, beta = 2)
```

[156]:

```
0.41995281429052916
```

[257]:

```
# y = pd.DataFrame(data=df_r['Churn'])
# X = df_r.loc[:, df_r.columns != 'Churn']
# y
```

[257]:

```
     Churn
0       No
1       No
2      Yes
3       No
4      Yes
…     …
7038    No
7039    No
7040    No
7041   Yes
7042    No

[7032 rows x 1 columns]
```

[13]:

```
y = df_r['Churn'].map(dict(Yes=1, No=0))
y
```

```
[13]: 0       0
      1       0
      2       1
      3       0
      4       1
             ..
      7038    0
      7039    0
      7040    0
      7041    1
      7042    0
      Name: Churn, Length: 7032, dtype: int64
```

```
[164]: from sklearn.pipeline import make_pipeline
       from sklearn.metrics import mean_squared_error
       from math import sqrt
       from sklearn.model_selection import GridSearchCV
       from sklearn.pipeline import make_pipeline
       from sklearn.compose import ColumnTransformer
       from sklearn.pipeline import Pipeline
       from sklearn.preprocessing import StandardScaler, OneHotEncoder,␣
        ↪OrdinalEncoder, LabelEncoder
       from sklearn.model_selection import train_test_split
```

```
[163]: ordinal_ftrs =␣
        ↪['PhoneService','MultipleLines','InternetService','OnlineSecurity',\
                       'OnlineBackup','DeviceProtection','TechSupport',\
                       'StreamingTV','StreamingMovies','Contract']
       ordinal_cats = [['No','Yes'],['No phone service','No','Yes'],['No','DSL','Fiber␣
        ↪optic'],['No internet service','No','Yes'],\
                        ['No internet service','No','Yes'],['No internet␣
        ↪service','No','Yes'],['No internet service','No','Yes'],\
                       ['No internet service','No','Yes'],['No internet␣
        ↪service','No','Yes'],['Month-to-month','One year','Two year']]


       cat_ftrs = ['gender', 'SeniorCitizen', 'Partner',␣
        ↪'Dependents','PaperlessBilling','PaymentMethod']


       num_ftrs = ['tenure','MonthlyCharges','TotalCharges']

       categorical_transformer = Pipeline(steps=[
           ('onehot', OneHotEncoder(sparse=False,handle_unknown='ignore'))])

       # ordinal encoder
       # We need to replace the NaN with a string first!
```

```
ordinal_transformer = Pipeline(steps=[
    ('ordinal', OrdinalEncoder(categories = ordinal_cats))])

# standard scaler
numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())])

# collect the transformers
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, num_ftrs),
        ('cat', categorical_transformer, cat_ftrs),
        ('ord', ordinal_transformer, ordinal_ftrs)])
```

[76]:

```
#  prep = Pipeline(steps=[('preprocessor', preprocessor)])
```

[411]:

```
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.model_selection import ParameterGrid

def MLpipe_Stratify_f1(X, y, preprocessor, ML_algo, param_grid):
    '''
    This function stratified-splits the data to training/validation/test (60/20/
 ↪20)
    The f1 score as metric score
    '''

    nr_states = 10
    test_scores = np.zeros(nr_states)
    val_best_scores = np.zeros(nr_states)
    final_models = []
    feature_importances = np.zeros(nr_states)

    for i in range(nr_states):

        print('\nrandoms state '+str(i+1))

        X_other, X_test, y_other, y_test = train_test_split(X,y,test_size = 0.
 ↪2,stratify = y, random_state=22*i)
        X_train, X_val, y_train, y_val =␣
 ↪train_test_split(X_other,y_other,test_size = 0.25, stratify = y_other,␣
 ↪random_state=22*i)

        train_score = np.zeros(len(ParameterGrid(param_grid)))
        val_score = np.zeros(len(ParameterGrid(param_grid)))
        X_train_prep = preprocessor.fit_transform(X_train)

#         feature_names = preprocessor.transformers_[0][-1] + \
#                 list(preprocessor.named_transformers_['cat'][0].
↪get_feature_names(cat_ftrs)) + \
#                 preprocessor.transformers_[2][-1]
#         print(feature_names)

        X_val_prep = preprocessor.transform(X_val)
        X_test_prep = preprocessor.transform(X_test)


        models = []

        for p in range(len(ParameterGrid(param_grid))):
            params = ParameterGrid(param_grid)[p]
            # print('   ',params)
            try:
                ML = ML_algo(random_state = 22*i)
            except:
                ML = ML_algo()

            ML.set_params(**params)
            ML.fit(X_train_prep,y_train)

            train_score[p] = f1_score(y_train, ML.predict(X_train_prep))

            models.append(ML)
            y_CV_pred = ML.predict(X_val_prep)
            val_score[p]= f1_score(y_val, y_CV_pred)

            # print('   ','train score:',train_score[p],'validation score:
↪',val_score[p])

        print([np.argmax(val_score)])
        print(models[np.argmax(val_score)])
        val_best_scores[i] = np.max(val_score)
        print('\nbest model parameters:',ParameterGrid(param_grid)[np.
↪argmax(val_score)])
        print('corresponding validation F1 score:',np.max(val_score))

        final_models.append(models[np.argmax(val_score)])

        y_test_pred = final_models[-1].predict(X_test_prep)
        test_scores[i] = f1_score(y_test, y_test_pred)

        print('test F1 score:',test_scores[i])

    return val_best_scores, test_scores, final_models, X_test_prep, y_test
```

[327]:

```
# Logistic Regression L1 Regularization

from sklearn.linear_model import LogisticRegression

params = { 'penalty' : ['l1'],
          'C' : [1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3, 1e4],
         'max_iter': [100000],
         'solver': ['saga'] }

Logl1_val_best_F1, Logl1_test_F1, Logl1_models = MLpipe_Stratify_f1(X, y, prep,␣
 ↪LogisticRegression, params)
```

```
randoms state 1
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.01}
    train score: 0.502247191011236 validation score: 0.521172638436482
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.1}
    train score: 0.5887016848364717 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1.0}
    train score: 0.6076071256620125 validation score: 0.5889046941678521
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10.0}
    train score: 0.6062650602409639 validation score: 0.5957446808510639
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 100.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1000.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10000.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
[5]
LogisticRegression(C=10.0, max_iter=100000, penalty='l1', random_state=0,
                   solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000,
'C': 10.0}
corresponding validation F1 score: 0.5957446808510639
test F1 score: 0.6169590643274854
```

```
randoms state 2
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.01}
    train score: 0.5105672969966629 validation score: 0.4966442953020133
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.1}
    train score: 0.5967342899554676 validation score: 0.5727272727272728
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1.0}
    train score: 0.6091173617846751 validation score: 0.573134328358209
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10.0}
    train score: 0.6084425036390102 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 100.0}
    train score: 0.6077669902912621 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
[5]
LogisticRegression(C=10.0, max_iter=100000, penalty='l1', random_state=22,
                   solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000,
'C': 10.0}
corresponding validation F1 score: 0.5819793205317577
test F1 score: 0.5889387144992526

randoms state 3
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.01}
    train score: 0.49577464788732395 validation score: 0.5083056478405316
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.1}
    train score: 0.5832502492522432 validation score: 0.6005747126436781
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1.0}
    train score: 0.591796875 validation score: 0.5895953757225434
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10.0}
    train score: 0.5941463414634146 validation score: 0.5902578796561604
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 100.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
[3]
```

```
LogisticRegression(C=0.1, max_iter=100000, penalty='l1', random_state=44,
                   solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000,
'C': 0.1}
corresponding validation F1 score: 0.6005747126436781
test F1 score: 0.5916795069337443

randoms state 4
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.01}
    train score: 0.5172413793103449 validation score: 0.5141955835962145
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.1}
    train score: 0.6015779092702169 validation score: 0.5839210155148097
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1.0}
    train score: 0.6069364161849711 validation score: 0.5885634588563459
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10.0}
    train score: 0.6094049904030711 validation score: 0.5905292479108636
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 100.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1000.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10000.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
[6]
LogisticRegression(C=100.0, max_iter=100000, penalty='l1', random_state=66,
                   solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000,
'C': 100.0}
corresponding validation F1 score: 0.592489568845619
test F1 score: 0.5915080527086385

randoms state 5
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.01}
    train score: 0.5107794361525705 validation score: 0.49423393739703464
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.1}
    train score: 0.5836228287841191 validation score: 0.5739910313901346
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1.0}
    train score: 0.6081474296799224 validation score: 0.5777777777777777
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10.0}
    train score: 0.6099565007249879 validation score: 0.5823529411764706
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 100.0}
    train score: 0.6102514506769826 validation score: 0.5832106038291606
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
[7]
LogisticRegression(C=1000.0, max_iter=100000, penalty='l1', random_state=88,
                   solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000,
'C': 1000.0}
corresponding validation F1 score: 0.5852941176470589
test F1 score: 0.5965417867435159

randoms state 6
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.01}
    train score: 0.4991587212563096 validation score: 0.5275459098497496
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.1}
    train score: 0.5946745562130178 validation score: 0.587183308494784
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1.0}
    train score: 0.6070565490575157 validation score: 0.5947521865889213
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10.0}
    train score: 0.6052123552123552 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 100.0}
    train score: 0.6061776061776061 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
[5]
LogisticRegression(C=10.0, max_iter=100000, penalty='l1', random_state=110,
                   solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000,
'C': 10.0}
corresponding validation F1 score: 0.6037735849056604
test F1 score: 0.6020260492040521

randoms state 7
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.01}
    train score: 0.5146005509641873 validation score: 0.48911222780569524
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.1}
    train score: 0.5853174603174602 validation score: 0.5735963581183612
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1.0}
    train score: 0.6064703042008691 validation score: 0.5877061469265367
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10.0}
    train score: 0.6081927710843373 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 100.0}
    train score: 0.6098265895953757 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1000.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10000.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
[5]
LogisticRegression(C=10.0, max_iter=100000, penalty='l1', random_state=132,
                   solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000,
'C': 10.0}
corresponding validation F1 score: 0.5898203592814371
test F1 score: 0.5982658959537572

randoms state 8
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.01}
    train score: 0.5245720596355604 validation score: 0.48344370860927144
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.1}
    train score: 0.6164451009354996 validation score: 0.5500747384155457
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1.0}
    train score: 0.6156843643448612 validation score: 0.5755813953488371
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10.0}
    train score: 0.6162847391516334 validation score: 0.5797101449275363
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 100.0}
    train score: 0.6153846153846154 validation score: 0.5797101449275363
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
[7]
LogisticRegression(C=1000.0, max_iter=100000, penalty='l1', random_state=154,
                   solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000,
'C': 1000.0}
corresponding validation F1 score: 0.5838150289017341
test F1 score: 0.5861561119293078

randoms state 9
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.01}
    train score: 0.4991587212563096 validation score: 0.4916387959866221
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.1}
    train score: 0.5879446640316206 validation score: 0.587887740029542
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1.0}
    train score: 0.5988372093023256 validation score: 0.6034985422740524
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10.0}
    train score: 0.6004842615012106 validation score: 0.6075581395348837
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 100.0}
    train score: 0.6005802707930368 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
[5]
LogisticRegression(C=10.0, max_iter=100000, penalty='l1', random_state=176,
                   solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000,
'C': 10.0}
corresponding validation F1 score: 0.6075581395348837
test F1 score: 0.5920471281296025

randoms state 10
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.01}
    train score: 0.5176211453744494 validation score: 0.4983498349834984
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 0.1}
    train score: 0.5954500494559841 validation score: 0.5697329376854601
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1.0}
    train score: 0.6085271317829458 validation score: 0.5840455840455839
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 100.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 1000.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000, 'C': 10000.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
[4]
LogisticRegression(max_iter=100000, penalty='l1', random_state=198,
                   solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100000,
'C': 1.0}
corresponding validation F1 score: 0.5840455840455839
test F1 score: 0.6
```

[868]:

```
# Saving Logistic L1 Regularization Best Models at 10 Random States
filename = dirct + '/results/log_l1_best_models.sav'
joblib.dump(Logl1_models, filename)
```

[868]:

```
['/Users/liyuetian1/Documents/GitHub/DATA1030_MidtermProject/results/log_l1_best
_models.sav']
```

[870]:

```
# Logistic Regression L2
params = { 'penalty' : ['l2'],
          'C' : [1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3, 1e4],
         'max_iter': [10000],
         'solver': ['saga'] }

Logl2_val_best_F1, Logl2_test_F1, Logl2_models, X_test, Y_test =␣
 ↪MLpipe_Stratify_f1(X, y, preprocessor, LogisticRegression, params)
```

```
randoms state 1
[5]
LogisticRegression(C=10.0, max_iter=10000, random_state=0, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 10.0}
corresponding validation F1 score: 0.5957446808510639
test F1 score: 0.6169590643274854

randoms state 2
[5]
LogisticRegression(C=10.0, max_iter=10000, random_state=22, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 10.0}
corresponding validation F1 score: 0.5819793205317577
test F1 score: 0.5889387144992526

randoms state 3
```

```
[3]
LogisticRegression(C=0.1, max_iter=10000, random_state=44, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 0.1}
corresponding validation F1 score: 0.5936599423631125
test F1 score: 0.5969230769230769

randoms state 4
[6]
LogisticRegression(C=100.0, max_iter=10000, random_state=66, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 100.0}
corresponding validation F1 score: 0.592489568845619
test F1 score: 0.5915080527086385

randoms state 5
[6]
LogisticRegression(C=100.0, max_iter=10000, random_state=88, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 100.0}
corresponding validation F1 score: 0.5852941176470589
test F1 score: 0.5965417867435159

randoms state 6
[5]
LogisticRegression(C=10.0, max_iter=10000, random_state=110, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 10.0}
corresponding validation F1 score: 0.6037735849056604
test F1 score: 0.6

randoms state 7
[4]
LogisticRegression(max_iter=10000, random_state=132, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 1.0}
corresponding validation F1 score: 0.5937031484257871
test F1 score: 0.5982658959537572

randoms state 8
[7]
LogisticRegression(C=1000.0, max_iter=10000, random_state=154, solver='saga')
```

```
best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 1000.0}
corresponding validation F1 score: 0.5838150289017341
test F1 score: 0.5861561119293078

randoms state 9
[4]
LogisticRegression(max_iter=10000, random_state=176, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 1.0}
corresponding validation F1 score: 0.6075581395348837
test F1 score: 0.5941176470588235

randoms state 10
[5]
LogisticRegression(C=10.0, max_iter=10000, random_state=198, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 10.0}
corresponding validation F1 score: 0.582857142857143
test F1 score: 0.5988372093023255
```

[871]:

```
# Saving Logistic L2 Regularization Best Models at 10 Random States
filename = dirct + '/results/log_l2_best_models.sav'
joblib.dump(Logl2_models, filename)
```

[871]:

```
['/Users/liyuetian1/Documents/GitHub/DATA1030_MidtermProject/results/log_l2_best
_models.sav']
```

[323]:

```
# Logistic Regression Elastic Net

params = { 'penalty' : ['elasticnet'],
           'C' : [1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3, 1e4],
          'l1_ratio': [0.01, 0.1, 0.25, 0.5, 0.75, 0.9, 0.99],
          'max_iter': [100000],
          'solver': ['saga'] }

LogEN_val_best_F1, LogEN_test_F1, LogEN_models = MLpipe_Stratify_f1(X, y,␣
 ↪preprocessor, LogisticRegression, params)
```

```
randoms state 1
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.0001}
```

```
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.001}
    train score: 0.18354430379746836 validation score: 0.2186046511627907
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.01}
    train score: 0.5609504132231405 validation score: 0.5606060606060607
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.01}
    train score: 0.5601249349297241 validation score: 0.5540334855403348
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.01}
    train score: 0.5487480021310602 validation score: 0.5412130637636081
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.01}
```

```
    train score: 0.5277015907844212 validation score: 0.5278219395866455
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.01}
    train score: 0.5139353400222966 validation score: 0.5283630470016207
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.01}
    train score: 0.5061728395061729 validation score: 0.5252854812398042
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.01}
    train score: 0.502247191011236 validation score: 0.521172638436482
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.1}
    train score: 0.6018563751831949 validation score: 0.5853658536585366
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.1}
    train score: 0.599706026457619 validation score: 0.5865522174535049
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.1}
    train score: 0.6013712047012733 validation score: 0.5882352941176471
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.1}
    train score: 0.6018654884634266 validation score: 0.5899280575539568
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.1}
    train score: 0.594381468703795 validation score: 0.5846599131693199
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.1}
    train score: 0.5898070262246413 validation score: 0.5846599131693199
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.1}
    train score: 0.5887016848364717 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1.0}
    train score: 0.6065573770491803 validation score: 0.59375
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1.0}
    train score: 0.6062650602409639 validation score: 0.59375
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1.0}
    train score: 0.6062650602409639 validation score: 0.5909090909090908
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1.0}
    train score: 0.6076071256620125 validation score: 0.5909090909090908
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1.0}
    train score: 0.6076071256620125 validation score: 0.5909090909090908
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1.0}
```

```
    train score: 0.6085700529610014 validation score: 0.5909090909090908
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1.0}
    train score: 0.6076071256620125 validation score: 0.5889046941678521
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10.0}
    train score: 0.6062650602409639 validation score: 0.5957446808510639
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10.0}
    train score: 0.6062650602409639 validation score: 0.5957446808510639
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10.0}
    train score: 0.6062650602409639 validation score: 0.5957446808510639
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10.0}
    train score: 0.6062650602409639 validation score: 0.5957446808510639
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10.0}
    train score: 0.6062650602409639 validation score: 0.5957446808510639
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10.0}
    train score: 0.6062650602409639 validation score: 0.5957446808510639
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10.0}
    train score: 0.6062650602409639 validation score: 0.5957446808510639
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 100.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 100.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 100.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 100.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 100.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 100.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 100.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1000.0}
```

```
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1000.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1000.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1000.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1000.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1000.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1000.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10000.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10000.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10000.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10000.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10000.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10000.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10000.0}
    train score: 0.6062650602409639 validation score: 0.594900849858357
[35]
LogisticRegression(C=10.0, l1_ratio=0.01, max_iter=100000, penalty='elasticnet',
                   random_state=0, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter':
100000, 'l1_ratio': 0.01, 'C': 10.0}
corresponding validation F1 score: 0.5957446808510639
test F1 score: 0.6169590643274854
```

```
randoms state 2
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.001}
    train score: 0.19370078740157484 validation score: 0.1375921375921376
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.01}
    train score: 0.5767060030785017 validation score: 0.5385826771653544
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
```

```
0.1, 'C': 0.01}
    train score: 0.5689210118740321 validation score: 0.5209003215434084
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.01}
    train score: 0.5601265822784809 validation score: 0.5244299674267101
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.01}
    train score: 0.53470715835141 validation score: 0.5074626865671642
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.01}
    train score: 0.525909592061742 validation score: 0.5041736227045076
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.01}
    train score: 0.5153032832498609 validation score: 0.5033333333333333
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.01}
    train score: 0.5103294249022893 validation score: 0.49916247906197664
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.1}
    train score: 0.6056751467710371 validation score: 0.5684210526315789
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.1}
    train score: 0.605288932419197 validation score: 0.5705705705705707
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.1}
    train score: 0.6031434184675836 validation score: 0.5748502994011976
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.1}
    train score: 0.6012814194184327 validation score: 0.5787106446776612
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.1}
    train score: 0.5995061728395061 validation score: 0.579185520361991
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.1}
    train score: 0.5985185185185184 validation score: 0.5783132530120482
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.1}
    train score: 0.597132970835393 validation score: 0.5770392749244713
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1.0}
    train score: 0.6104651162790697 validation score: 0.5786350148367952
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1.0}
    train score: 0.6107610276296654 validation score: 0.5765230312035662
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1.0}
    train score: 0.6107610276296654 validation score: 0.5756676557863502
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
```

```
0.5, 'C': 1.0}
    train score: 0.6091173617846751 validation score: 0.5777777777777777
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1.0}
    train score: 0.6087378640776699 validation score: 0.5786350148367952
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1.0}
    train score: 0.6091173617846751 validation score: 0.5773809523809523
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1.0}
    train score: 0.6091173617846751 validation score: 0.573134328358209
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10.0}
    train score: 0.6084425036390102 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10.0}
    train score: 0.6084425036390102 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10.0}
    train score: 0.6084425036390102 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10.0}
    train score: 0.6084425036390102 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10.0}
    train score: 0.6084425036390102 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10.0}
    train score: 0.6084425036390102 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10.0}
    train score: 0.6084425036390102 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 100.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 100.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 100.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 100.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 100.0}
    train score: 0.6077669902912621 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
```

```
0.9, 'C': 100.0}
    train score: 0.6077669902912621 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 100.0}
    train score: 0.6077669902912621 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10000.0}
    train score: 0.6074721009218826 validation score: 0.5819793205317577
```

[35]

```
LogisticRegression(C=10.0, l1_ratio=0.01, max_iter=100000, penalty='elasticnet',
                   random_state=22, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter':
100000, 'l1_ratio': 0.01, 'C': 10.0}
corresponding validation F1 score: 0.5819793205317577
test F1 score: 0.5889387144992526

randoms state 3
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.001}
    train score: 0.15384615384615385 validation score: 0.11793611793611795
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.01}
    train score: 0.5507853403141362 validation score: 0.5535168195718654
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.01}
    train score: 0.5472794506075014 validation score: 0.5432098765432098
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.01}
    train score: 0.534048257372654 validation score: 0.5339652448657187
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.01}
    train score: 0.5219298245614036 validation score: 0.5281803542673108
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.01}
    train score: 0.5102834908282379 validation score: 0.5154975530179445
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.01}
    train score: 0.5036537380550871 validation score: 0.5114754098360657
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.01}
    train score: 0.49577464788732395 validation score: 0.5083056478405316
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.1}
    train score: 0.5828797624938149 validation score: 0.5936599423631125
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.1}
    train score: 0.5821782178217821 validation score: 0.5936599423631125
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.1}
    train score: 0.5829195630585898 validation score: 0.5931232091690544
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.1}
    train score: 0.5819631290483308 validation score: 0.5974395448079659
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.1}
    train score: 0.5850746268656716 validation score: 0.592274678111588
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.1}
    train score: 0.5835411471321695 validation score: 0.599713055954089
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.1}
    train score: 0.5832502492522432 validation score: 0.6005747126436781
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1.0}
    train score: 0.593460224499756 validation score: 0.5919540229885057
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1.0}
    train score: 0.593460224499756 validation score: 0.5899280575539568
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1.0}
    train score: 0.593460224499756 validation score: 0.5870503597122301
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1.0}
    train score: 0.5930630190522717 validation score: 0.5887445887445887
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1.0}
    train score: 0.5933528836754642 validation score: 0.5895953757225434
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1.0}
    train score: 0.5920859794821691 validation score: 0.5895953757225434
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1.0}
    train score: 0.591796875 validation score: 0.5895953757225434
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10.0}
    train score: 0.5941463414634146 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10.0}
    train score: 0.5941463414634146 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10.0}
    train score: 0.5941463414634146 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10.0}
    train score: 0.5941463414634146 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10.0}
    train score: 0.5941463414634146 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10.0}
    train score: 0.5941463414634146 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10.0}
    train score: 0.5941463414634146 validation score: 0.5902578796561604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 100.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 100.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 100.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 100.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 100.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 100.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 100.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10000.0}
    train score: 0.5938566552901025 validation score: 0.5894134477825466
[27]
LogisticRegression(C=0.1, l1_ratio=0.99, max_iter=100000, penalty='elasticnet',
                   random_state=44, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter':
100000, 'l1_ratio': 0.99, 'C': 0.1}
corresponding validation F1 score: 0.6005747126436781
test F1 score: 0.5916795069337443

randoms state 4
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.001}
    train score: 0.19607843137254902 validation score: 0.21495327102803738
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.001}
```

```
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.01}
    train score: 0.574083634486319 validation score: 0.5542168674698795
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.01}
    train score: 0.5689027561102444 validation score: 0.5460030165912519
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.01}
    train score: 0.5585585585585585 validation score: 0.5482388973966309
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.01}
    train score: 0.5326027397260273 validation score: 0.5360501567398119
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.01}
    train score: 0.5193370165745858 validation score: 0.5266457680250783
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.01}
    train score: 0.5177777777777778 validation score: 0.5220125786163522
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.01}
    train score: 0.5172413793103449 validation score: 0.5141955835962145
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.1}
    train score: 0.6038104543234002 validation score: 0.5907172995780591
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.1}
    train score: 0.6051732552464617 validation score: 0.5875706214689266
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.1}
    train score: 0.6029411764705883 validation score: 0.5875706214689266
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.1}
    train score: 0.6036256736893679 validation score: 0.5864022662889519
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.1}
    train score: 0.6034398034398034 validation score: 0.5867418899858956
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.1}
```

```
    train score: 0.6034398034398034 validation score: 0.5859154929577466
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.1}
    train score: 0.6015779092702169 validation score: 0.5839210155148097
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1.0}
    train score: 0.6073147256977862 validation score: 0.5885634588563459
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1.0}
    train score: 0.6073147256977862 validation score: 0.5885634588563459
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1.0}
    train score: 0.6073147256977862 validation score: 0.5885634588563459
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1.0}
    train score: 0.6066441983630236 validation score: 0.5885634588563459
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1.0}
    train score: 0.6072289156626506 validation score: 0.5885634588563459
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1.0}
    train score: 0.6069364161849711 validation score: 0.5885634588563459
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1.0}
    train score: 0.6069364161849711 validation score: 0.5885634588563459
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10.0}
    train score: 0.6094049904030711 validation score: 0.5905292479108636
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10.0}
    train score: 0.6094049904030711 validation score: 0.5905292479108636
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10.0}
    train score: 0.6100719424460431 validation score: 0.5905292479108636
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10.0}
    train score: 0.6100719424460431 validation score: 0.5905292479108636
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10.0}
    train score: 0.6094049904030711 validation score: 0.5905292479108636
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10.0}
    train score: 0.6094049904030711 validation score: 0.5905292479108636
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10.0}
    train score: 0.6094049904030711 validation score: 0.5905292479108636
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 100.0}
```

```
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 100.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 100.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 100.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 100.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 100.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 100.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1000.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1000.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1000.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1000.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1000.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1000.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1000.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10000.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10000.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10000.0}
```

```
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10000.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10000.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10000.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10000.0}
    train score: 0.6097794822627037 validation score: 0.592489568845619
[42]
LogisticRegression(C=100.0, l1_ratio=0.01, max_iter=100000,
                   penalty='elasticnet', random_state=66, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter':
100000, 'l1_ratio': 0.01, 'C': 100.0}
corresponding validation F1 score: 0.592489568845619
test F1 score: 0.5915080527086385

randoms state 5
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.001}
    train score: 0.18383518225039622 validation score: 0.18527315914489312
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
```

```
0.1, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.01}
    train score: 0.5622739018087856 validation score: 0.5598755832037324
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.01}
    train score: 0.5567765567765568 validation score: 0.5460317460317461
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.01}
    train score: 0.5458399576046635 validation score: 0.5399361022364216
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.01}
    train score: 0.5342019543973943 validation score: 0.5194805194805195
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.01}
    train score: 0.5168788046485888 validation score: 0.5090311986863709
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.01}
    train score: 0.5108273181565798 validation score: 0.5
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.01}
    train score: 0.5086254869226489 validation score: 0.49586776859504134
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.1}
    train score: 0.5932872655478777 validation score: 0.5769805680119581
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.1}
    train score: 0.5925925925925927 validation score: 0.5761194029850746
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.1}
    train score: 0.5935802469135801 validation score: 0.5765765765765766
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
```

```
0.5, 'C': 0.1}
    train score: 0.5899209486166008 validation score: 0.5748502994011976
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.1}
    train score: 0.5893385982230998 validation score: 0.5739910313901346
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.1}
    train score: 0.5841584158415841 validation score: 0.5748502994011976
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.1}
    train score: 0.5843253968253969 validation score: 0.5739910313901346
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1.0}
    train score: 0.6067961165048543 validation score: 0.5798816568047337
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1.0}
    train score: 0.6074721009218826 validation score: 0.5798816568047337
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1.0}
    train score: 0.6065891472868218 validation score: 0.5798816568047337
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1.0}
    train score: 0.6078526417838099 validation score: 0.5798816568047337
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1.0}
    train score: 0.6074721009218826 validation score: 0.5798816568047337
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1.0}
    train score: 0.6081474296799224 validation score: 0.5777777777777777
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1.0}
    train score: 0.6081474296799224 validation score: 0.5777777777777777
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10.0}
    train score: 0.6105466860183842 validation score: 0.5832106038291606
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10.0}
    train score: 0.6105466860183842 validation score: 0.5832106038291606
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10.0}
    train score: 0.6105466860183842 validation score: 0.5832106038291606
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10.0}
    train score: 0.6102514506769826 validation score: 0.5823529411764706
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10.0}
    train score: 0.6102514506769826 validation score: 0.5823529411764706
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
```

```
0.9, 'C': 10.0}
    train score: 0.6099565007249879 validation score: 0.5823529411764706
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10.0}
    train score: 0.6099565007249879 validation score: 0.5823529411764706
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 100.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 100.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 100.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 100.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 100.0}
    train score: 0.6102514506769826 validation score: 0.5832106038291606
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 100.0}
    train score: 0.6102514506769826 validation score: 0.5832106038291606
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 100.0}
    train score: 0.6102514506769826 validation score: 0.5832106038291606
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
```

```
0.01, 'C': 10000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10000.0}
    train score: 0.6105466860183842 validation score: 0.5852941176470589
[42]
LogisticRegression(C=100.0, l1_ratio=0.01, max_iter=100000,
                   penalty='elasticnet', random_state=88, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter':
100000, 'l1_ratio': 0.01, 'C': 100.0}
corresponding validation F1 score: 0.5852941176470589
test F1 score: 0.5965417867435159

randoms state 6
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.001}
    train score: 0.16051364365971107 validation score: 0.18138424821002389
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.01}
    train score: 0.5621454357916451 validation score: 0.571875
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.01}
    train score: 0.5580426861009891 validation score: 0.5687203791469194
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.01}
    train score: 0.5428265524625266 validation score: 0.5654952076677315
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.01}
    train score: 0.5206384149697303 validation score: 0.5415986949429037
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.01}
    train score: 0.50917176209005 validation score: 0.5250836120401338
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.01}
    train score: 0.5047459519821329 validation score: 0.53
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.01}
    train score: 0.5033482142857143 validation score: 0.5291181364392677
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.1}
    train score: 0.5965601965601965 validation score: 0.5891016200294551
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.1}
    train score: 0.5975429975429974 validation score: 0.5941176470588235
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.1}
    train score: 0.5969563082965145 validation score: 0.591715976331361
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.1}
    train score: 0.5981308411214954 validation score: 0.5929203539823009
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.1}
    train score: 0.5973385904386398 validation score: 0.5829596412556053
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.1}
    train score: 0.5936883629191321 validation score: 0.5868263473053892
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.1}
    train score: 0.5936883629191321 validation score: 0.587183308494784
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1.0}
    train score: 0.6077294685990339 validation score: 0.5997088791848617
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1.0}
    train score: 0.6077294685990339 validation score: 0.5997088791848617
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1.0}
    train score: 0.6077294685990339 validation score: 0.5997088791848617
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1.0}
    train score: 0.6070565490575157 validation score: 0.5976676384839649
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1.0}
    train score: 0.6073500967117988 validation score: 0.5967976710334788
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1.0}
    train score: 0.6070565490575157 validation score: 0.5947521865889213
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1.0}
    train score: 0.6070565490575157 validation score: 0.5947521865889213
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10.0}
    train score: 0.6048309178743961 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10.0}
    train score: 0.6051232479458675 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10.0}
    train score: 0.6051232479458675 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10.0}
    train score: 0.6048309178743961 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10.0}
    train score: 0.6052123552123552 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10.0}
    train score: 0.6052123552123552 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10.0}
    train score: 0.6052123552123552 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 100.0}
    train score: 0.6057971014492753 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 100.0}
    train score: 0.6057971014492753 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 100.0}
    train score: 0.6057971014492753 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 100.0}
    train score: 0.6057971014492753 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 100.0}
    train score: 0.6057971014492753 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 100.0}
    train score: 0.6055045871559633 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 100.0}
    train score: 0.6061776061776061 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10000.0}
    train score: 0.6064703042008691 validation score: 0.6037735849056604
[35]
LogisticRegression(C=10.0, l1_ratio=0.01, max_iter=100000, penalty='elasticnet',
                   random_state=110, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter':
100000, 'l1_ratio': 0.01, 'C': 10.0}
corresponding validation F1 score: 0.6037735849056604
test F1 score: 0.6

randoms state 7
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.0001}
```

```
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.001}
    train score: 0.18354430379746836 validation score: 0.1686746987951807
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.01}
    train score: 0.5633074935400516 validation score: 0.5468998410174881
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.01}
    train score: 0.5617860851505712 validation score: 0.5408
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.01}
    train score: 0.5514316012725344 validation score: 0.5492730210016156
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.01}
    train score: 0.5345572354211662 validation score: 0.506578947368421
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.01}
    train score: 0.5289617486338798 validation score: 0.4966887417218543
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.01}
```

```
    train score: 0.520065970313359 validation score: 0.4983388704318937
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.01}
    train score: 0.5146005509641873 validation score: 0.48911222780569524
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.1}
    train score: 0.5970588235294118 validation score: 0.592814371257485
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.1}
    train score: 0.5973516429622364 validation score: 0.592814371257485
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.1}
    train score: 0.5984251968503937 validation score: 0.5873493975903614
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.1}
    train score: 0.5956607495069034 validation score: 0.5873493975903614
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.1}
    train score: 0.592482690405539 validation score: 0.5864661654135338
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.1}
    train score: 0.5902125556104796 validation score: 0.5735963581183612
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.1}
    train score: 0.5856079404466501 validation score: 0.5714285714285714
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1.0}
    train score: 0.6061776061776061 validation score: 0.5937031484257871
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1.0}
    train score: 0.6061776061776061 validation score: 0.5919282511210762
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1.0}
    train score: 0.6064703042008691 validation score: 0.5877061469265367
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1.0}
    train score: 0.6074360212457751 validation score: 0.5877061469265367
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1.0}
    train score: 0.6061776061776061 validation score: 0.5877061469265367
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1.0}
    train score: 0.6061776061776061 validation score: 0.5877061469265367
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1.0}
    train score: 0.6064703042008691 validation score: 0.5877061469265367
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10.0}
```

```
    train score: 0.6098265895953757 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10.0}
    train score: 0.6098265895953757 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10.0}
    train score: 0.6095329802599905 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10.0}
    train score: 0.6095329802599905 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10.0}
    train score: 0.6081927710843373 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10.0}
    train score: 0.6081927710843373 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10.0}
    train score: 0.6081927710843373 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 100.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 100.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 100.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 100.0}
    train score: 0.6098265895953757 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 100.0}
    train score: 0.6098265895953757 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 100.0}
    train score: 0.6098265895953757 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 100.0}
    train score: 0.6098265895953757 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1000.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1000.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1000.0}
```

```
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1000.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1000.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1000.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1000.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10000.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10000.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10000.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10000.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10000.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10000.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10000.0}
    train score: 0.6091566265060242 validation score: 0.5898203592814371
[28]
LogisticRegression(l1_ratio=0.01, max_iter=100000, penalty='elasticnet',
                   random_state=132, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter':
100000, 'l1_ratio': 0.01, 'C': 1.0}
corresponding validation F1 score: 0.5937031484257871
test F1 score: 0.5982658959537572

randoms state 8
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
```

```
0.1, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.001}
    train score: 0.20109976433621368 validation score: 0.15130023640661938
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.01}
    train score: 0.5837615621788282 validation score: 0.5195618153364632
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.01}
    train score: 0.5767634854771784 validation score: 0.5087440381558028
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.01}
    train score: 0.5655391120507399 validation score: 0.5080385852090032
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
```

```
0.5, 'C': 0.01}
    train score: 0.5428881650380022 validation score: 0.5008130081300813
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.01}
    train score: 0.5345113197128658 validation score: 0.4943089430894309
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.01}
    train score: 0.5277161862527716 validation score: 0.4868421052631579
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.01}
    train score: 0.5245720596355604 validation score: 0.48344370860927144
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.1}
    train score: 0.6128873585833744 validation score: 0.5663716814159293
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.1}
    train score: 0.6131889763779528 validation score: 0.5663716814159293
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.1}
    train score: 0.6141732283464566 validation score: 0.5650887573964498
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.1}
    train score: 0.6151574803149606 validation score: 0.5621301775147929
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.1}
    train score: 0.613714849531327 validation score: 0.5557206537890045
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.1}
    train score: 0.6153846153846154 validation score: 0.5514157973174367
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.1}
    train score: 0.6164451009354996 validation score: 0.5500747384155457
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1.0}
    train score: 0.6159844054580896 validation score: 0.5726744186046511
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1.0}
    train score: 0.6159844054580896 validation score: 0.5726744186046511
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1.0}
    train score: 0.6159844054580896 validation score: 0.5726744186046511
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1.0}
    train score: 0.6159102000976086 validation score: 0.5718432510885341
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1.0}
    train score: 0.615609756097561 validation score: 0.5735080058224162
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
```

```
0.9, 'C': 1.0}
    train score: 0.6159844054580896 validation score: 0.5735080058224162
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1.0}
    train score: 0.6156843643448612 validation score: 0.5755813953488371
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10.0}
    train score: 0.6159844054580896 validation score: 0.5797101449275363
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10.0}
    train score: 0.6159844054580896 validation score: 0.5797101449275363
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10.0}
    train score: 0.6159844054580896 validation score: 0.5797101449275363
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10.0}
    train score: 0.6162847391516334 validation score: 0.5797101449275363
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10.0}
    train score: 0.6162847391516334 validation score: 0.5797101449275363
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10.0}
    train score: 0.6162847391516334 validation score: 0.5797101449275363
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10.0}
    train score: 0.6162847391516334 validation score: 0.5797101449275363
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 100.0}
    train score: 0.6160583941605839 validation score: 0.5817655571635311
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 100.0}
    train score: 0.6160583941605839 validation score: 0.5817655571635311
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 100.0}
    train score: 0.6160583941605839 validation score: 0.5817655571635311
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 100.0}
    train score: 0.6160583941605839 validation score: 0.5817655571635311
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 100.0}
    train score: 0.6153846153846154 validation score: 0.5797101449275363
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 100.0}
    train score: 0.6153846153846154 validation score: 0.5797101449275363
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 100.0}
    train score: 0.6153846153846154 validation score: 0.5797101449275363
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
```

```
0.01, 'C': 1000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10000.0}
    train score: 0.6160583941605839 validation score: 0.5838150289017341
[49]
LogisticRegression(C=1000.0, l1_ratio=0.01, max_iter=100000,
                   penalty='elasticnet', random_state=154, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter':
100000, 'l1_ratio': 0.01, 'C': 1000.0}
corresponding validation F1 score: 0.5838150289017341
```

```
test F1 score: 0.5861561119293078

randoms state 9
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.001}
    train score: 0.1774960380348653 validation score: 0.14457831325301204
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.01}
    train score: 0.561494551115724 validation score: 0.5670731707317073
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.01}
    train score: 0.5547981122181437 validation score: 0.5687789799072644
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.01}
    train score: 0.5344735435595938 validation score: 0.5615141955835963
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.01}
    train score: 0.5189248491497532 validation score: 0.5138211382113822
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.01}
    train score: 0.5094339622641509 validation score: 0.4966666666666666
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.01}
    train score: 0.501952035694367 validation score: 0.4916387959866221
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.01}
    train score: 0.4991587212563096 validation score: 0.4916387959866221
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.1}
    train score: 0.5933528836754642 validation score: 0.5985185185185184
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.1}
    train score: 0.5928466438020579 validation score: 0.5985185185185184
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.1}
    train score: 0.5914664051005394 validation score: 0.5976331360946746
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.1}
    train score: 0.5887573964497042 validation score: 0.5949926362297496
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.1}
    train score: 0.5901477832512316 validation score: 0.5911764705882353
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.1}
    train score: 0.5897435897435896 validation score: 0.5899705014749264
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.1}
    train score: 0.5879446640316206 validation score: 0.587887740029542
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1.0}
    train score: 0.5991274842462433 validation score: 0.6075581395348837
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1.0}
    train score: 0.5991274842462433 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1.0}
    train score: 0.5981580222976249 validation score: 0.6034985422740524
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1.0}
    train score: 0.5984481086323958 validation score: 0.6034985422740524
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1.0}
    train score: 0.5974781765276429 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1.0}
    train score: 0.5988372093023256 validation score: 0.6034985422740524
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1.0}
    train score: 0.5988372093023256 validation score: 0.6034985422740524
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10.0}
    train score: 0.6008708272859217 validation score: 0.6075581395348837
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10.0}
    train score: 0.5999032414126754 validation score: 0.6075581395348837
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10.0}
    train score: 0.6001936108422071 validation score: 0.6075581395348837
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10.0}
    train score: 0.6001936108422071 validation score: 0.6075581395348837
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10.0}
    train score: 0.6004842615012106 validation score: 0.6075581395348837
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10.0}
    train score: 0.6004842615012106 validation score: 0.6075581395348837
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10.0}
    train score: 0.6004842615012106 validation score: 0.6075581395348837
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 100.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 100.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 100.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 100.0}
    train score: 0.6005802707930368 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 100.0}
    train score: 0.6005802707930368 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 100.0}
    train score: 0.6005802707930368 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 100.0}
    train score: 0.6005802707930368 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10000.0}
    train score: 0.5999032414126754 validation score: 0.6055312954876273
```

```
[28]
LogisticRegression(l1_ratio=0.01, max_iter=100000, penalty='elasticnet',
                   random_state=176, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter':
100000, 'l1_ratio': 0.01, 'C': 1.0}
corresponding validation F1 score: 0.6075581395348837
test F1 score: 0.5941176470588235

randoms state 10
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.0001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.001}
    train score: 0.18196202531645572 validation score: 0.1674641148325359
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.001}
```

```
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.001}
    train score: 0.0 validation score: 0.0
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.01}
    train score: 0.5674273858921163 validation score: 0.5454545454545454
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.01}
    train score: 0.566839378238342 validation score: 0.5352112676056339
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.01}
    train score: 0.5577227200843436 validation score: 0.5161290322580646
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.01}
    train score: 0.53470715835141 validation score: 0.5057096247960848
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.01}
    train score: 0.5263157894736842 validation score: 0.5
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.01}
    train score: 0.519580805295091 validation score: 0.5
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.01}
    train score: 0.5176211453744494 validation score: 0.4983498349834984
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 0.1}
    train score: 0.604197169350903 validation score: 0.577259475218659
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 0.1}
    train score: 0.6013712047012733 validation score: 0.5789473684210527
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 0.1}
    train score: 0.6000978952520802 validation score: 0.5797950219619327
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 0.1}
    train score: 0.5998043052837574 validation score: 0.5747800586510263
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 0.1}
    train score: 0.5950738916256156 validation score: 0.5726872246696034
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 0.1}
    train score: 0.5952615992102666 validation score: 0.5731166912850812
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 0.1}
    train score: 0.5951557093425606 validation score: 0.5718518518518518
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1.0}
```

```
    train score: 0.6092843326885881 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1.0}
    train score: 0.6095791001451378 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1.0}
    train score: 0.6095791001451378 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1.0}
    train score: 0.6101694915254238 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1.0}
    train score: 0.6092009685230024 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1.0}
    train score: 0.6085271317829458 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1.0}
    train score: 0.6085271317829458 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10.0}
    train score: 0.6073500967117988 validation score: 0.582857142857143
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10.0}
    train score: 0.6070565490575157 validation score: 0.582857142857143
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 100.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 100.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 100.0}
```

```
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 100.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 100.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 100.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 100.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 1000.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 1000.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 1000.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 1000.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 1000.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 1000.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 1000.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.01, 'C': 10000.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.1, 'C': 10000.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.25, 'C': 10000.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.5, 'C': 10000.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.75, 'C': 10000.0}
```

```
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.9, 'C': 10000.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
    {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter': 100000, 'l1_ratio':
0.99, 'C': 10000.0}
    train score: 0.6070565490575157 validation score: 0.5820256776034237
[35]
LogisticRegression(C=10.0, l1_ratio=0.01, max_iter=100000, penalty='elasticnet',
                   random_state=198, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'elasticnet', 'max_iter':
100000, 'l1_ratio': 0.01, 'C': 10.0}
corresponding validation F1 score: 0.582857142857143
test F1 score: 0.5988372093023255
```

```
[872]: # Saving Logistic Elastic Net Best Models at 10 Random States
       filename = dirct + '/results/log_EN_best_models.sav'
       joblib.dump(LogEN_models, filename)
```

```
[872]: ['/Users/liyuetian1/Documents/GitHub/DATA1030_MidtermProject/results/log_EN_best
       _models.sav']
```

```
[342]: # SVC
       from sklearn.svm import SVC

       params = {
           'gamma' : [1e-2, 1e-1, 1e0, 1e1, 1e2, 'auto', 'scale'],
           'C': [0.01, 0.1, 0.5, 1, 5, 10, 20]
       }

       SVC_val_best_F1, SVC_test_F1, SVC_models = MLpipe_Stratify_f1(X, y,␣
        ↪preprocessor, SVC, params)
```

```
randoms state 1
[20]
SVC(C=0.5, random_state=0)

best model parameters: {'gamma': 'scale', 'C': 0.5}
corresponding validation F1 score: 0.5861027190332326
test F1 score: 0.5834633385335413

randoms state 2
[34]
SVC(C=5, random_state=22)

best model parameters: {'gamma': 'scale', 'C': 5}
```

```
corresponding validation F1 score: 0.554858934169279
test F1 score: 0.5578446909667196

randoms state 3
[34]
SVC(C=5, random_state=44)

best model parameters: {'gamma': 'scale', 'C': 5}
corresponding validation F1 score: 0.5892857142857142
test F1 score: 0.5795275590551182

randoms state 4
[35]
SVC(C=10, gamma=0.01, random_state=66)

best model parameters: {'gamma': 0.01, 'C': 10}
corresponding validation F1 score: 0.5710059171597633
test F1 score: 0.5628930817610063

randoms state 5
[29]
SVC(C=5, gamma=0.1, random_state=88)

best model parameters: {'gamma': 0.1, 'C': 5}
corresponding validation F1 score: 0.5910447761194031
test F1 score: 0.573082489146165

randoms state 6
[34]
SVC(C=5, random_state=110)

best model parameters: {'gamma': 'scale', 'C': 5}
corresponding validation F1 score: 0.5885885885885886
test F1 score: 0.5592705167173253

randoms state 7
[29]
SVC(C=5, gamma=0.1, random_state=132)

best model parameters: {'gamma': 0.1, 'C': 5}
corresponding validation F1 score: 0.5585585585585586
test F1 score: 0.5756676557863502

randoms state 8
[41]
SVC(C=10, random_state=154)

best model parameters: {'gamma': 'scale', 'C': 10}
```

```
corresponding validation F1 score: 0.5567010309278351
test F1 score: 0.5666666666666668

randoms state 9
[34]
SVC(C=5, random_state=176)

best model parameters: {'gamma': 'scale', 'C': 5}
corresponding validation F1 score: 0.5785609397944199
test F1 score: 0.5603715170278637

randoms state 10
[47]
SVC(C=20, gamma='auto', random_state=198)

best model parameters: {'gamma': 'auto', 'C': 20}
corresponding validation F1 score: 0.5641791044776119
test F1 score: 0.5722983257229831
```

[873]:

```
# Saving SVC Best Models at 10 Random States

filename = dirct + '/results/SVC_best_models.sav'
joblib.dump(SVC_models, filename)
```

[873]:

```
['/Users/liyuetian1/Documents/GitHub/DATA1030_MidtermProject/results/SVC_best_mo
dels.sav']
```

[340]:

```
# KNN

from sklearn.neighbors import KNeighborsClassifier

params = {
    'n_neighbors': [1, 2, 3, 5, 10, 30, 50, 100, 200],
    'weights': ['uniform', 'distance']
         }

KNN_val_best_F1, KNN_test_F1, KNN_models = MLpipe_Stratify_f1(X, y,␣
 ↪preprocessor, KNeighborsClassifier, params)
```

```
randoms state 1
[17]
KNeighborsClassifier(n_neighbors=200, weights='distance')

best model parameters: {'weights': 'distance', 'n_neighbors': 200}
corresponding validation F1 score: 0.5888594164456235
test F1 score: 0.5897079276773296
```

```
randoms state 2
[10]
KNeighborsClassifier(n_neighbors=30)

best model parameters: {'weights': 'uniform', 'n_neighbors': 30}
corresponding validation F1 score: 0.568888888888889
test F1 score: 0.5901162790697674

randoms state 3
[17]
KNeighborsClassifier(n_neighbors=200, weights='distance')

best model parameters: {'weights': 'distance', 'n_neighbors': 200}
corresponding validation F1 score: 0.5875862068965517
test F1 score: 0.6092124814264487

randoms state 4
[12]
KNeighborsClassifier(n_neighbors=50)

best model parameters: {'weights': 'uniform', 'n_neighbors': 50}
corresponding validation F1 score: 0.5969738651994497
test F1 score: 0.5760233918128655

randoms state 5
[12]
KNeighborsClassifier(n_neighbors=50)

best model parameters: {'weights': 'uniform', 'n_neighbors': 50}
corresponding validation F1 score: 0.593886462882096
test F1 score: 0.5932203389830508

randoms state 6
[17]
KNeighborsClassifier(n_neighbors=200, weights='distance')

best model parameters: {'weights': 'distance', 'n_neighbors': 200}
corresponding validation F1 score: 0.6
test F1 score: 0.5694249649368864

randoms state 7
[16]
KNeighborsClassifier(n_neighbors=200)

best model parameters: {'weights': 'uniform', 'n_neighbors': 200}
corresponding validation F1 score: 0.6008583690987125
test F1 score: 0.5573294629898403
```

```
randoms state 8
[14]
KNeighborsClassifier(n_neighbors=100)

best model parameters: {'weights': 'uniform', 'n_neighbors': 100}
corresponding validation F1 score: 0.5751072961373391
test F1 score: 0.5714285714285715

randoms state 9
[12]
KNeighborsClassifier(n_neighbors=50)

best model parameters: {'weights': 'uniform', 'n_neighbors': 50}
corresponding validation F1 score: 0.6067415730337079
test F1 score: 0.5965417867435159

randoms state 10
[17]
KNeighborsClassifier(n_neighbors=200, weights='distance')

best model parameters: {'weights': 'distance', 'n_neighbors': 200}
corresponding validation F1 score: 0.5726256983240224
test F1 score: 0.6102635228848821
```

[874]:

```python
# Saving KNN Best Models at 10 Random States

filename = dirct + '/results/KNN_best_models.sav'
joblib.dump(KNN_models, filename)
```

[874]:

```
['/Users/liyuetian1/Documents/GitHub/DATA1030_MidtermProject/results/KNN_best_mo
dels.sav']
```

[341]:

```python
# Random Forest

from sklearn.ensemble import RandomForestClassifier

params = { 'max_features': [1, 3, 5, 10, 20, None],
         'max_depth':  [1, 3, 5, 7, 10, 15, 20, None]}
        # 'min_samples_split': [ 2, 5, 10, 15, 25, 30] }

RF_val_best_F1, RF_test_F1, RF_models = MLpipe_Stratify_f1(X, y, preprocessor,␣
 ↪RandomForestClassifier, params)
```

```
randoms state 1
[138]
RandomForestClassifier(max_depth=7, max_features=None, random_state=0)
```

```
best model parameters: {'min_samples_split': 2, 'max_features': None,
'max_depth': 7}
corresponding validation F1 score: 0.596045197740113
test F1 score: 0.5964391691394659

randoms state 2
[192]
RandomForestClassifier(max_depth=15, max_features=5, random_state=22)

best model parameters: {'min_samples_split': 2, 'max_features': 5, 'max_depth':
15}
corresponding validation F1 score: 0.5792507204610953
test F1 score: 0.5814307458143075

randoms state 3
[121]
RandomForestClassifier(max_depth=7, max_features=5, min_samples_split=5,
                       random_state=44)

best model parameters: {'min_samples_split': 5, 'max_features': 5, 'max_depth':
7}
corresponding validation F1 score: 0.5852187028657617
test F1 score: 0.577922077922078

randoms state 4
[132]
RandomForestClassifier(max_depth=7, max_features=20, random_state=66)

best model parameters: {'min_samples_split': 2, 'max_features': 20, 'max_depth':
7}
corresponding validation F1 score: 0.5891016200294551
test F1 score: 0.5740458015267177

randoms state 5
[130]
RandomForestClassifier(max_depth=7, max_features=10, min_samples_split=25,
                       random_state=88)

best model parameters: {'min_samples_split': 25, 'max_features': 10,
'max_depth': 7}
corresponding validation F1 score: 0.5899705014749264
test F1 score: 0.5944363103953147

randoms state 6
[209]
RandomForestClassifier(max_depth=15, max_features=20, min_samples_split=30,
                       random_state=110)
```

```
best model parameters: {'min_samples_split': 30, 'max_features': 20,
'max_depth': 15}
corresponding validation F1 score: 0.6156069364161849
test F1 score: 0.5692995529061102

randoms state 7
[142]
RandomForestClassifier(max_depth=7, max_features=None, min_samples_split=25,
                       random_state=132)

best model parameters: {'min_samples_split': 25, 'max_features': None,
'max_depth': 7}
corresponding validation F1 score: 0.5786350148367952
test F1 score: 0.5718518518518518

randoms state 8
[158]
RandomForestClassifier(max_depth=10, max_features=5, min_samples_split=10,
                       random_state=154)

best model parameters: {'min_samples_split': 10, 'max_features': 5, 'max_depth':
10}
corresponding validation F1 score: 0.562406015037594
test F1 score: 0.576271186440678

randoms state 9
[127]
RandomForestClassifier(max_depth=7, max_features=10, min_samples_split=5,
                       random_state=176)

best model parameters: {'min_samples_split': 5, 'max_features': 10, 'max_depth':
7}
corresponding validation F1 score: 0.5889387144992526
test F1 score: 0.5875190258751902

randoms state 10
[170]
RandomForestClassifier(max_depth=10, max_features=20, min_samples_split=10,
                       random_state=198)

best model parameters: {'min_samples_split': 10, 'max_features': 20,
'max_depth': 10}
corresponding validation F1 score: 0.5681159420289854
test F1 score: 0.5892857142857142
```

[875]:

```
# Saving RF Best Models at 10 Random States
```

```
filename = dirct + '/results/RF_best_models.sav'
joblib.dump(RF_models, filename)
```

[875]:
```
['/Users/liyuetian1/Documents/GitHub/DATA1030_MidtermProject/results/RF_best_mod
els.sav']
```

[359]:
```
# Xgboost
from sklearn.model_selection import ParameterGrid
import xgboost

param_grid = {
      "learning_rate": [0.05, 0.1, 0.2, 0.3],
      "n_estimators": [1000],
      "seed": [0],
      "gamma": [0, 0.1, 0.2, 0.3, 0.4],
      "colsample_bytree": [0.3, 0.4, 0.5, 0.7],
      "subsample": [0.4, 0.5, 0.65, 0.75, 1],
      "min_child_weight": [1, 3, 5, 7],
      "eval_metric": ['logloss']
       }

nr_states = 10

XBG_test_scores = np.zeros(nr_states)
XGB_val_best_scores = np.zeros(nr_states)
XGB_final_models = []
XGB_feature_importances = np.zeros(nr_states)

for i in range(nr_states):

    models = []

    print('\nrandoms state '+str(i+1))

    X_other, X_test, y_other, y_test = train_test_split(X,y,test_size = 0.
 ↪2,stratify = y, random_state=22*i)
    X_train, X_val, y_train, y_val = train_test_split(X_other,y_other,test_size␣
 ↪= 0.25, stratify = y_other, random_state=22*i)

    ## Preprocess
    X_train_prep = preprocessor.fit_transform(X_train)
    X_val_prep = preprocessor.transform(X_val)
    X_test_prep = preprocessor.transform(X_test)

    XGB_feature_names = preprocessor.transformers_[0][-1] + \
                list(preprocessor.named_transformers_['cat'][0].
 ↪get_feature_names(cat_ftrs)) + \
                preprocessor.transformers_[2][-1]

    train_score = np.zeros(len(ParameterGrid(param_grid)))
    val_score = np.zeros(len(ParameterGrid(param_grid)))

    ## Loop through parameters
    for p in range(len(ParameterGrid(param_grid))):
        params = ParameterGrid(param_grid)[p]
        XGB = xgboost.XGBClassifier(use_label_encoder =False, random_state =␣
↪22*i)
        XGB.set_params(**params)
        XGB.fit(X_train_prep,y_train,early_stopping_rounds=50,␣
↪eval_set=[(X_val_prep, y_val)], verbose=False)

        models.append(XGB) # save it
        y_val_pred = XGB.predict(X_val_prep)

        val_score[p] = f1_score(y_val, y_val_pred)

    XGB_val_best_scores = np.max(val_score)

    print('\nbest model parameters:',ParameterGrid(param_grid)[np.
↪argmax(val_score)])
    print('corresponding validation score:',np.max(val_score))

    XGB_final_models.append(models[np.argmax(val_score)])

    y_test_pred = XGB_final_models[-1].predict(X_test_prep)

    # calculate and save the test score
    XBG_test_scores[i] = f1_score(y_test, y_test_pred)

    print('test f1 score:', XBG_test_scores[i])
```

```
randoms state 1

best model parameters: {'subsample': 0.4, 'seed': 0, 'n_estimators': 1000,
'min_child_weight': 1, 'learning_rate': 0.2, 'gamma': 0.2, 'eval_metric':
'logloss', 'colsample_bytree': 0.5}
corresponding validation score: 0.6031294452347084
test f1 score: 0.5882352941176471

randoms state 2

best model parameters: {'subsample': 0.5, 'seed': 0, 'n_estimators': 1000,
'min_child_weight': 7, 'learning_rate': 0.3, 'gamma': 0.3, 'eval_metric':
'logloss', 'colsample_bytree': 0.3}
corresponding validation score: 0.5937031484257871
test f1 score: 0.5727554179566563

randoms state 3

best model parameters: {'subsample': 0.4, 'seed': 0, 'n_estimators': 1000,
'min_child_weight': 5, 'learning_rate': 0.2, 'gamma': 0.3, 'eval_metric':
'logloss', 'colsample_bytree': 0.3}
corresponding validation score: 0.6023391812865497
test f1 score: 0.5750000000000001

randoms state 4

best model parameters: {'subsample': 0.65, 'seed': 0, 'n_estimators': 1000,
'min_child_weight': 3, 'learning_rate': 0.3, 'gamma': 0.1, 'eval_metric':
'logloss', 'colsample_bytree': 0.7}
corresponding validation score: 0.6054519368723099
test f1 score: 0.5630498533724341

randoms state 5

best model parameters: {'subsample': 1, 'seed': 0, 'n_estimators': 1000,
'min_child_weight': 5, 'learning_rate': 0.3, 'gamma': 0.1, 'eval_metric':
'logloss', 'colsample_bytree': 0.3}
corresponding validation score: 0.5994152046783625
test f1 score: 0.6051873198847263

randoms state 6

best model parameters: {'subsample': 0.65, 'seed': 0, 'n_estimators': 1000,
'min_child_weight': 3, 'learning_rate': 0.1, 'gamma': 0.1, 'eval_metric':
'logloss', 'colsample_bytree': 0.7}
corresponding validation score: 0.6251808972503619
test f1 score: 0.5837037037037036

randoms state 7

best model parameters: {'subsample': 0.4, 'seed': 0, 'n_estimators': 1000,
'min_child_weight': 5, 'learning_rate': 0.3, 'gamma': 0.4, 'eval_metric':
'logloss', 'colsample_bytree': 0.5}
corresponding validation score: 0.6002886002886003
test f1 score: 0.6025824964131994

randoms state 8

best model parameters: {'subsample': 0.4, 'seed': 0, 'n_estimators': 1000,
'min_child_weight': 5, 'learning_rate': 0.2, 'gamma': 0.3, 'eval_metric':
'logloss', 'colsample_bytree': 0.7}
corresponding validation score: 0.5852941176470589
test f1 score: 0.5612403100775193

randoms state 9

best model parameters: {'subsample': 0.75, 'seed': 0, 'n_estimators': 1000,
'min_child_weight': 1, 'learning_rate': 0.3, 'gamma': 0.3, 'eval_metric':
'logloss', 'colsample_bytree': 0.5}
corresponding validation score: 0.6075581395348837
test f1 score: 0.5843373493975903

randoms state 10

best model parameters: {'subsample': 0.5, 'seed': 0, 'n_estimators': 1000,
'min_child_weight': 1, 'learning_rate': 0.3, 'gamma': 0.4, 'eval_metric':
'logloss', 'colsample_bytree': 0.3}
corresponding validation score: 0.5906432748538012
test f1 score: 0.5825825825825826
```

[877]:

```
# Saving RF Best Models at 10 Random States

filename = dirct + '/results/XGBoost_best_models.sav'
joblib.dump(XGB_final_models, filename)
```

[877]:

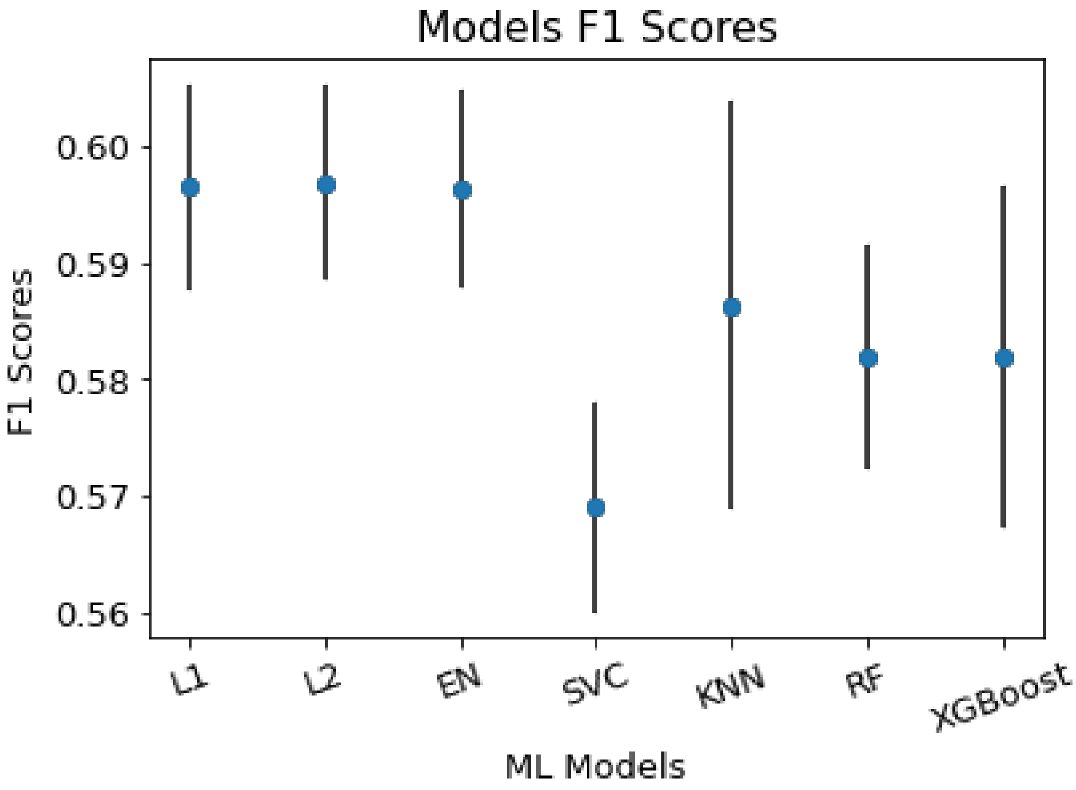
```
['/Users/liyuetian1/Documents/GitHub/DATA1030_MidtermProject/results/XGBoost_bes
t_models.sav']
```

3 3. Results

[884]:

```
# Models F1 Scores Summary

rand_states = np.arange(1,11,1)

column_names = ['L1','L2','EN','SVC', 'KNN','RF','XGBoost']

models_F1 = pd.DataFrame([Logl1_test_F1, Logl2_test_F1, LogEN_test_F1,␣
 ↪SVC_test_F1, KNN_test_F1, RF_test_F1, XBG_test_scores], index=column_names,␣
 ↪columns = rand_states)

models_F1 = models_F1.T

models_F1
```

```
mean_F1 = models_F1.mean(axis=0)
std_F1 = models_F1.std(axis=0)


plt.figure()
plt.rcParams.update({'font.size': 13})
plt.plot(column_names, mean_F1 , 'o', color= 'k')
plt.errorbar(column_names, mean_F1, yerr= std_F1, fmt='o', capsize= 0.5,␣
 ↪capthick= 0.2, ecolor='k')

plt.ylabel('F1 Scores')
plt.xlabel('ML Models')
plt.title('Models F1 Scores')
plt.xticks(rotation=20)
plt.savefig(dirct +'/figures/Models_F1_Scores.png',dpi=300)
plt.show()
```



[364]:
```
mean_F1 = models_F1.mean(axis=0)
print(mean_F1)
```

[364]:
```
Lasso      0.596412
Ridge      0.596825
EN         0.596300
SVC        0.569109
KNN        0.586327
RF         0.581850
XGBoost    0.581867
dtype: float64
```

[879]:
```
print(std_F1)
```

```
Lasso      0.008797
Ridge      0.008388
EN         0.008544
SVC        0.008995
KNN        0.017413
RF         0.009567
XGBoost    0.014656
dtype: float64
```

Decide on Logistic L2 Regularization Model

4 4. Model Interpretation

[474]:
```
# Pipeline for Logistic Regression L2 Regularization Only
```

[885]:
```
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.model_selection import ParameterGrid

def Logl2_Stratify_f1(X, y, preprocessor, ML_algo, param_grid, verbose = 1):
    '''
    This function intends to focus on analyzing and interpretating the final␣
 ↪model
    Logistic Regression L2 Regularization;
    - Same random states, parameters as previous one
    - Additional outputs for feature importance calculation
    '''

    nr_states = 10
    test_scores = np.zeros(nr_states)
    val_best_scores = np.zeros(nr_states)
    final_models = []
    feature_importances = np.zeros(nr_states)
    X_test_all = []
    Y_test_all = []


    for i in range(nr_states):

        print('\nrandoms state '+str(i+1))

        X_other, X_test, y_other, y_test = train_test_split(X,y,test_size = 0.
↪2,stratify = y, random_state=22*i)
        X_train, X_val, y_train, y_val =␣
↪train_test_split(X_other,y_other,test_size = 0.25, stratify = y_other,␣
↪random_state=22*i)

        train_score = np.zeros(len(ParameterGrid(param_grid)))
        val_score = np.zeros(len(ParameterGrid(param_grid)))

        X_train_prep = preprocessor.fit_transform(X_train)

        feature_names = preprocessor.transformers_[0][-1] + \
                list(preprocessor.named_transformers_['cat'][0].
↪get_feature_names(cat_ftrs)) + \
                preprocessor.transformers_[2][-1]

        X_val_prep = preprocessor.transform(X_val)
        X_test_prep = preprocessor.transform(X_test)


        if verbose ==2 :
            final_scaler = StandardScaler()
            X_train_prep = final_scaler.fit_transform(X_train_prep)
            X_val_prep = final_scaler.transform(X_val_prep)
            X_test_prep = final_scaler.transform(X_test_prep)
            print('Mean Standardized All Features')


        X_test_all.append(X_test_prep)
        Y_test_all.append(y_test)

        models = []

        for p in range(len(ParameterGrid(param_grid))):
            params = ParameterGrid(param_grid)[p]
            try:
                ML = ML_algo(random_state = 22*i)
            except:
                ML = ML_algo()
            ML.set_params(**params)
            ML.fit(X_train_prep,y_train)

            train_score[p] = f1_score(y_train, ML.predict(X_train_prep))

            models.append(ML)
            y_CV_pred = ML.predict(X_val_prep)
            val_score[p]= f1_score(y_val, y_CV_pred)

        print([np.argmax(val_score)])
        print(models[np.argmax(val_score)])
        val_best_scores[i] = np.max(val_score)
        print('\nbest model parameters:',ParameterGrid(param_grid)[np.
 ↪argmax(val_score)])
        print('corresponding validation F1 score:',np.max(val_score))

        final_models.append(models[np.argmax(val_score)])

        y_test_pred = final_models[-1].predict(X_test_prep)


        test_scores[i] = f1_score(y_test, y_test_pred)

        print('test F1 score:',test_scores[i])

    return val_best_scores, test_scores, final_models, X_test_all, Y_test_all,␣
 ↪feature_names
```

[890]:

```
# Logistic Regression l2
## Verbose = 1, no final mean standardization

params = { 'penalty' : ['l2'],
          'C' : [1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3, 1e4],
         'max_iter': [10000],
         'solver': ['saga'] }

Logl2_val_best_F1, Logl2_test_F1_v1, Logl2_models_v1, all_X_test_v1,␣
 ↪all_Y_test_v1, feature_names = Logl2_Stratify_f1(X, y, preprocessor,␣
 ↪LogisticRegression, params, 1)
```

```
randoms state 1
[5]
LogisticRegression(C=10.0, max_iter=10000, random_state=0, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 10.0}
corresponding validation F1 score: 0.5957446808510639
test F1 score: 0.6169590643274854

randoms state 2
[5]
LogisticRegression(C=10.0, max_iter=10000, random_state=22, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 10.0}
corresponding validation F1 score: 0.5819793205317577
test F1 score: 0.5889387144992526

randoms state 3
[3]
LogisticRegression(C=0.1, max_iter=10000, random_state=44, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 0.1}
corresponding validation F1 score: 0.5936599423631125
test F1 score: 0.5969230769230769

randoms state 4
[6]
LogisticRegression(C=100.0, max_iter=10000, random_state=66, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 100.0}
corresponding validation F1 score: 0.592489568845619
test F1 score: 0.5915080527086385

randoms state 5
[6]
LogisticRegression(C=100.0, max_iter=10000, random_state=88, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 100.0}
corresponding validation F1 score: 0.5852941176470589
test F1 score: 0.5965417867435159

randoms state 6
[5]
LogisticRegression(C=10.0, max_iter=10000, random_state=110, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 10.0}
corresponding validation F1 score: 0.6037735849056604
test F1 score: 0.6
```

```
randoms state 7
[4]
LogisticRegression(max_iter=10000, random_state=132, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 1.0}
corresponding validation F1 score: 0.5937031484257871
test F1 score: 0.5982658959537572

randoms state 8
[7]
LogisticRegression(C=1000.0, max_iter=10000, random_state=154, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 1000.0}
corresponding validation F1 score: 0.5838150289017341
test F1 score: 0.5861561119293078

randoms state 9
[4]
LogisticRegression(max_iter=10000, random_state=176, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 1.0}
corresponding validation F1 score: 0.6075581395348837
test F1 score: 0.5941176470588235

randoms state 10
[5]
LogisticRegression(C=10.0, max_iter=10000, random_state=198, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 10.0}
corresponding validation F1 score: 0.582857142857143
test F1 score: 0.5988372093023255
```

4.0.1 4.1 Global Feature Importance

4.1.1 Permutation Feature Importance

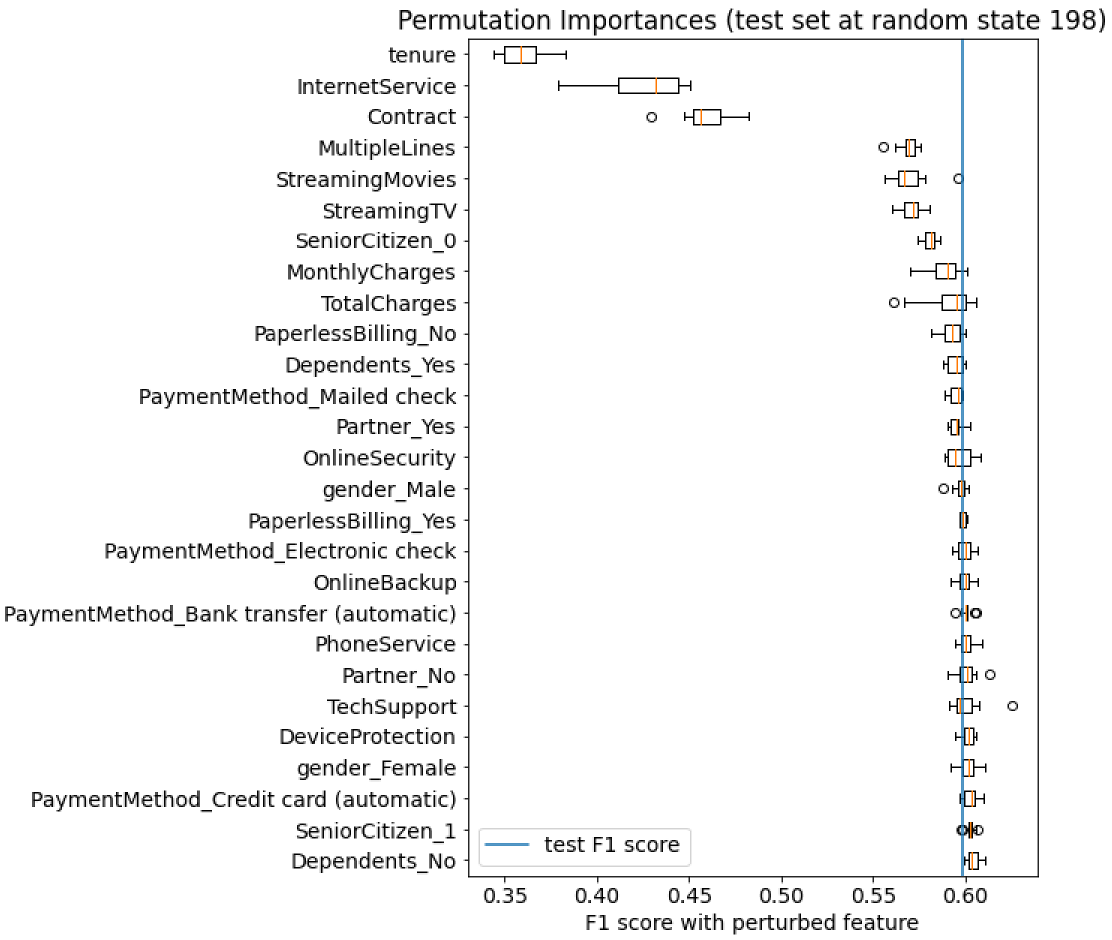
[891]:

```
# Choose the Last Model from 10 Random States, C = 10.0



model = Logl2_models_v1[-1]

X_test_df = pd.DataFrame(data=all_X_test_v1[-1], columns=feature_names)
Y_test = all_Y_test_v1[-1]
```

```
np.random.seed(42)

nr_runs = 10
scores = np.zeros([len(feature_names),nr_runs])

for i in range(len(feature_names)):
    print('shuffling '+str(feature_names[i]))
    f1_scores = []
    for j in range(nr_runs):
        X_test_shuffled = X_test_df.copy()
        X_test_shuffled[feature_names[i]] = np.random.
 ↪permutation(X_test_df[feature_names[i]].values)
        perm_Y_test_pred = model.predict(X_test_shuffled)

        f1_scores.append(f1_score(Y_test,perm_Y_test_pred))
    print('   shuffled test score:',np.around(np.mean(f1_scores),3),'+/-',np.
 ↪around(np.std(f1_scores),3))
    scores[i] = f1_scores

sorted_indcs = np.argsort(np.mean(scores,axis=1))[::-1]

plt.rcParams.update({'font.size': 14})
plt.figure(figsize=(10,9))

label_ft = [feature_names[i] for i in sorted_indcs]

plt.boxplot(scores[sorted_indcs].T, labels= label_ft , vert=False)
plt.axvline(Logl2_test_F1_v1[-1],label='test F1 score')
plt.title("Permutation Importances (test set at random state 198)")
plt.xlabel('F1 score with perturbed feature')
plt.legend()
plt.tight_layout()
plt.savefig(dirct +'/figures/PermutationImportances_LogL2.png',dpi=300)
plt.show()
```

```
shuffling tenure
   shuffled test score: 0.359 +/- 0.012
shuffling MonthlyCharges
   shuffled test score: 0.588 +/- 0.008
shuffling TotalCharges
   shuffled test score: 0.59 +/- 0.014
shuffling gender_Female
   shuffled test score: 0.601 +/- 0.005
shuffling gender_Male
   shuffled test score: 0.597 +/- 0.004
```

```
shuffling SeniorCitizen_0
   shuffled test score: 0.581 +/- 0.004
shuffling SeniorCitizen_1
   shuffled test score: 0.603 +/- 0.003
shuffling Partner_No
   shuffled test score: 0.601 +/- 0.006
shuffling Partner_Yes
   shuffled test score: 0.595 +/- 0.004
shuffling Dependents_No
   shuffled test score: 0.604 +/- 0.004
shuffling Dependents_Yes
   shuffled test score: 0.594 +/- 0.004
shuffling PaperlessBilling_No
   shuffled test score: 0.592 +/- 0.005
shuffling PaperlessBilling_Yes
   shuffled test score: 0.599 +/- 0.002
shuffling PaymentMethod_Bank transfer (automatic)
   shuffled test score: 0.6 +/- 0.003
shuffling PaymentMethod_Credit card (automatic)
   shuffled test score: 0.602 +/- 0.004
shuffling PaymentMethod_Electronic check
   shuffled test score: 0.599 +/- 0.004
shuffling PaymentMethod_Mailed check
   shuffled test score: 0.595 +/- 0.003
shuffling PhoneService
   shuffled test score: 0.601 +/- 0.004
shuffling MultipleLines
   shuffled test score: 0.569 +/- 0.006
shuffling InternetService
   shuffled test score: 0.425 +/- 0.023
shuffling OnlineSecurity
   shuffled test score: 0.597 +/- 0.007
shuffling OnlineBackup
   shuffled test score: 0.6 +/- 0.004
shuffling DeviceProtection
   shuffled test score: 0.601 +/- 0.004
shuffling TechSupport
   shuffled test score: 0.601 +/- 0.009
shuffling StreamingTV
   shuffled test score: 0.571 +/- 0.007
shuffling StreamingMovies
   shuffled test score: 0.57 +/- 0.011
shuffling Contract
   shuffled test score: 0.458 +/- 0.014
```



4.0.2 4.1.2 Coefficients

```
[777]: # Verbose = 2, mean standardize all features to calculate scaled coefficients

params = { 'penalty' : ['l2'],
           'C' : [1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3, 1e4],
          'max_iter': [10000],
          'solver': ['saga'] }

Logl2_val_best_F1, Logl2_test_F1, Logl2_models_v2, all_X_test_v2,␣
 ↪all_Y_test_v2, feature_names = Logl2_Stratify_f1(X, y, preprocessor,␣
 ↪LogisticRegression, params, 2)
```

```
randoms state 1
Mean Standardized All Features
[4]
LogisticRegression(max_iter=10000, random_state=0, solver='saga')
```

```
best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 1.0}
corresponding validation F1 score: 0.5968882602545968
test F1 score: 0.6218978102189782

randoms state 2
Mean Standardized All Features
[5]
LogisticRegression(C=10.0, max_iter=10000, random_state=22, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 10.0}
corresponding validation F1 score: 0.5819793205317577
test F1 score: 0.5880597014925374

randoms state 3
Mean Standardized All Features
[5]
LogisticRegression(C=10.0, max_iter=10000, random_state=44, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 10.0}
corresponding validation F1 score: 0.5894134477825466
test F1 score: 0.6079027355623101

randoms state 4
Mean Standardized All Features
[3]
LogisticRegression(C=0.1, max_iter=10000, random_state=66, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 0.1}
corresponding validation F1 score: 0.5938375350140056
test F1 score: 0.5920471281296025

randoms state 5
Mean Standardized All Features
[6]
LogisticRegression(C=100.0, max_iter=10000, random_state=88, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 100.0}
corresponding validation F1 score: 0.5852941176470589
test F1 score: 0.5965417867435159

randoms state 6
Mean Standardized All Features
```

```
[5]
LogisticRegression(C=10.0, max_iter=10000, random_state=110, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 10.0}
corresponding validation F1 score: 0.6037735849056604
test F1 score: 0.6

randoms state 7
Mean Standardized All Features
[4]
LogisticRegression(max_iter=10000, random_state=132, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 1.0}
corresponding validation F1 score: 0.5910447761194031
test F1 score: 0.5991316931982633

randoms state 8
Mean Standardized All Features
[6]
LogisticRegression(C=100.0, max_iter=10000, random_state=154, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 100.0}
corresponding validation F1 score: 0.5838150289017341
test F1 score: 0.5861561119293078

randoms state 9
Mean Standardized All Features
[4]
LogisticRegression(max_iter=10000, random_state=176, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 1.0}
corresponding validation F1 score: 0.6055312954876273
test F1 score: 0.5941176470588235

randoms state 10
Mean Standardized All Features
[4]
LogisticRegression(max_iter=10000, random_state=198, solver='saga')

best model parameters: {'solver': 'saga', 'penalty': 'l2', 'max_iter': 10000,
'C': 1.0}
corresponding validation F1 score: 0.5840455840455839
test F1 score: 0.6040462427745665
```

[778]:
```
df = pd.DataFrame(columns=feature_names)

for i in range(len(Logl2_models_v2)):
    coeffs = Logl2_models_v2[i].coef_
    coeffs = coeffs.flatten()
    df.loc[len(df)] = coeffs

df
```

[778]:
```
     tenure  MonthlyCharges  TotalCharges  gender_Female  gender_Male  \
0 -1.455692       -0.485221      0.717505       0.006505    -0.006505
1 -1.413223       -0.448889      0.653662       0.016476    -0.016476
2 -1.287476       -0.196579      0.572477       0.007304    -0.007304
3 -1.109718        0.028232      0.300720       0.005105    -0.005105
4 -1.400814       -0.508331      0.628667      -0.001046     0.001046
5 -1.722746       -0.592581      0.993347      -0.016378     0.016378
6 -1.446682       -0.483506      0.734251       0.010144    -0.010144
7 -1.624607       -0.151269      0.813818       0.001865    -0.001865
8 -1.352395       -0.481333      0.577631       0.015167    -0.015167
9 -1.410448       -0.540946      0.654020      -0.012774     0.012774

   SeniorCitizen_0  SeniorCitizen_1  Partner_No  Partner_Yes  Dependents_No  \
0        -0.038321         0.038321    0.010419    -0.010419       0.014601
1        -0.037852         0.037852    0.005839    -0.005839       0.028027
2        -0.025366         0.025366   -0.013296     0.013296       0.052123
3        -0.055956         0.055956   -0.023226     0.023226       0.065793
4        -0.033812         0.033812    0.002221    -0.002221       0.037855
5        -0.044401         0.044401   -0.004297     0.004297       0.014468
6        -0.048017         0.048017    0.028825    -0.028825       0.002484
7        -0.035895         0.035895   -0.003849     0.003849       0.037616
8        -0.041493         0.041493    0.003832    -0.003832       0.051598
9        -0.023258         0.023258    0.016375    -0.016375       0.044045

   …  PhoneService  MultipleLines  InternetService  OnlineSecurity  \
0  …     -0.119622       0.139245         0.832630       -0.184672
1  …     -0.179400       0.245220         0.866304       -0.248488
2  …     -0.304697       0.213082         0.729479       -0.260861
3  …     -0.333867       0.169113         0.744774       -0.311952
4  …     -0.208066       0.264237         0.952716       -0.209141
5  …     -0.180441       0.195462         0.969734       -0.239209
6  …     -0.154598       0.182867         0.937884       -0.278372
7  …     -0.237823       0.140958         0.730981       -0.250504
8  …     -0.187144       0.223166         0.864561       -0.164183
9  …     -0.249395       0.263084         0.978672       -0.189443

   OnlineBackup  DeviceProtection  TechSupport  StreamingTV  StreamingMovies  \
0     -0.132269          0.074815    -0.263076     0.296624         0.254979
```

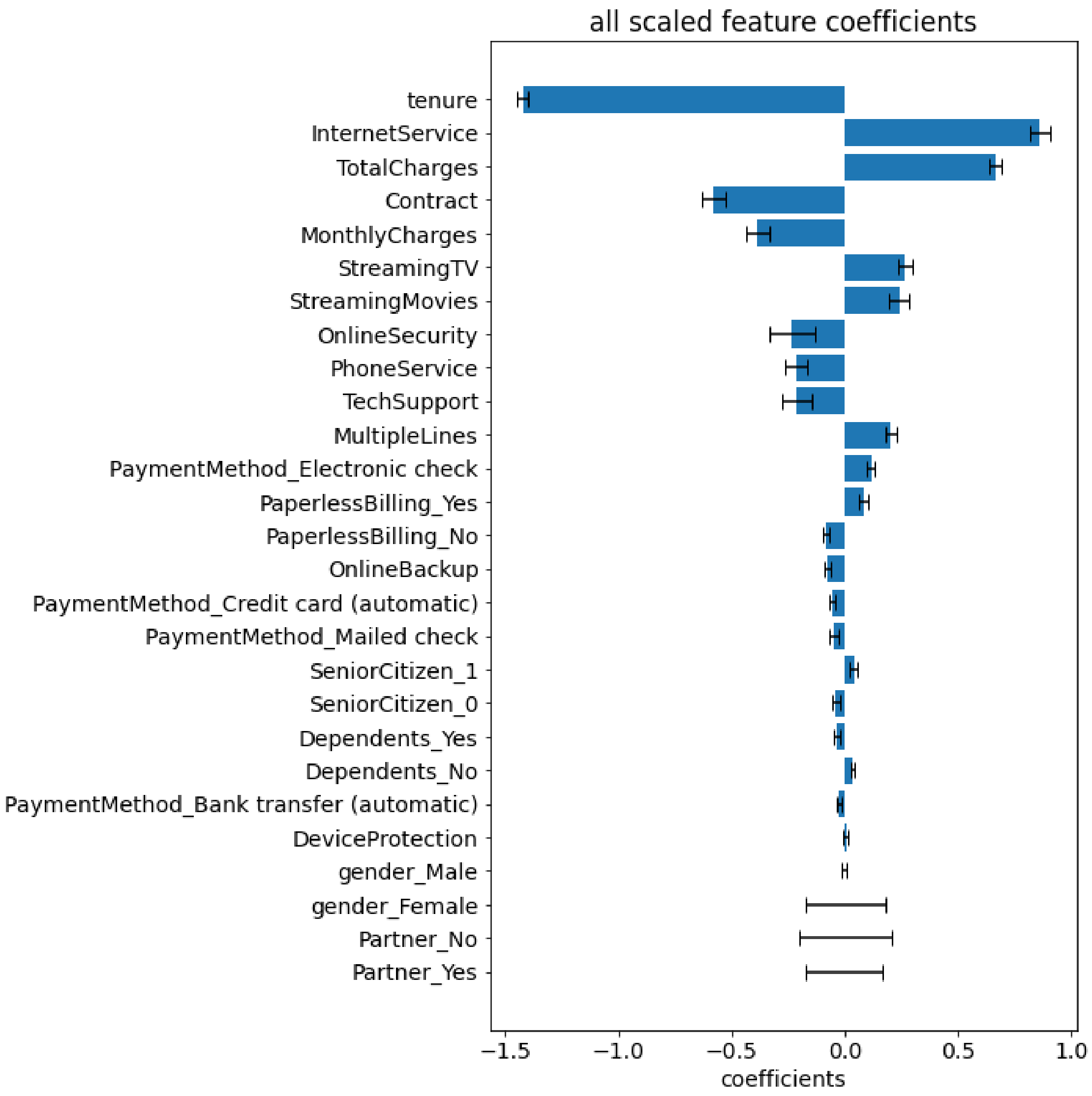
```
1     -0.104634          0.056365    -0.220938     0.281135          0.273190
2     -0.057228          0.043276    -0.277963     0.226318          0.245375
3     -0.082721         -0.044520    -0.135700     0.230002          0.131058
4     -0.048219          0.019298    -0.222804     0.276559          0.214199
5     -0.039704         -0.030931    -0.208715     0.261747          0.260644
6     -0.069856         -0.049884    -0.193395     0.301565          0.281122
7     -0.109290         -0.038859    -0.143211     0.222316          0.220335
8     -0.103697          0.078021    -0.273035     0.300060          0.253790
9     -0.043081         -0.044265    -0.197872     0.268159          0.271240

   Contract
0 -0.605205
1 -0.602587
2 -0.576734
3 -0.552380
4 -0.585439
5 -0.594631
6 -0.568038
7 -0.539228
8 -0.547004
9 -0.617027

[10 rows x 27 columns]
```

[779]:

```
cof_mean = df.mean(axis = 0)
cof_std = df.std(axis = 0)
```

[780]:

```
sorted_indcs = np.argsort(np.abs(cof_mean))

FN_coef = [feature_names[i] for i in sorted_indcs[-27:]]
FN_coef

plt.figure(figsize=(10,10))

plt.rcParams.update({'font.size': 14})
plt.barh(np.arange(27),cof_mean[sorted_indcs[-27:]], xerr = cof_std, ␣
 ↪align='center', alpha=1.0, ecolor='black', capsize=5)
plt.yticks(np.arange(27),FN_coef)
plt.xlabel('coefficients')
plt.title('all scaled feature coefficients')
plt.tight_layout()
plt.savefig(dirct +'/figures/LR_coefs_scaled.png',dpi=300)
plt.show()
```



4.1.3 Using SHAP to calcualte global feature importance

[893]:

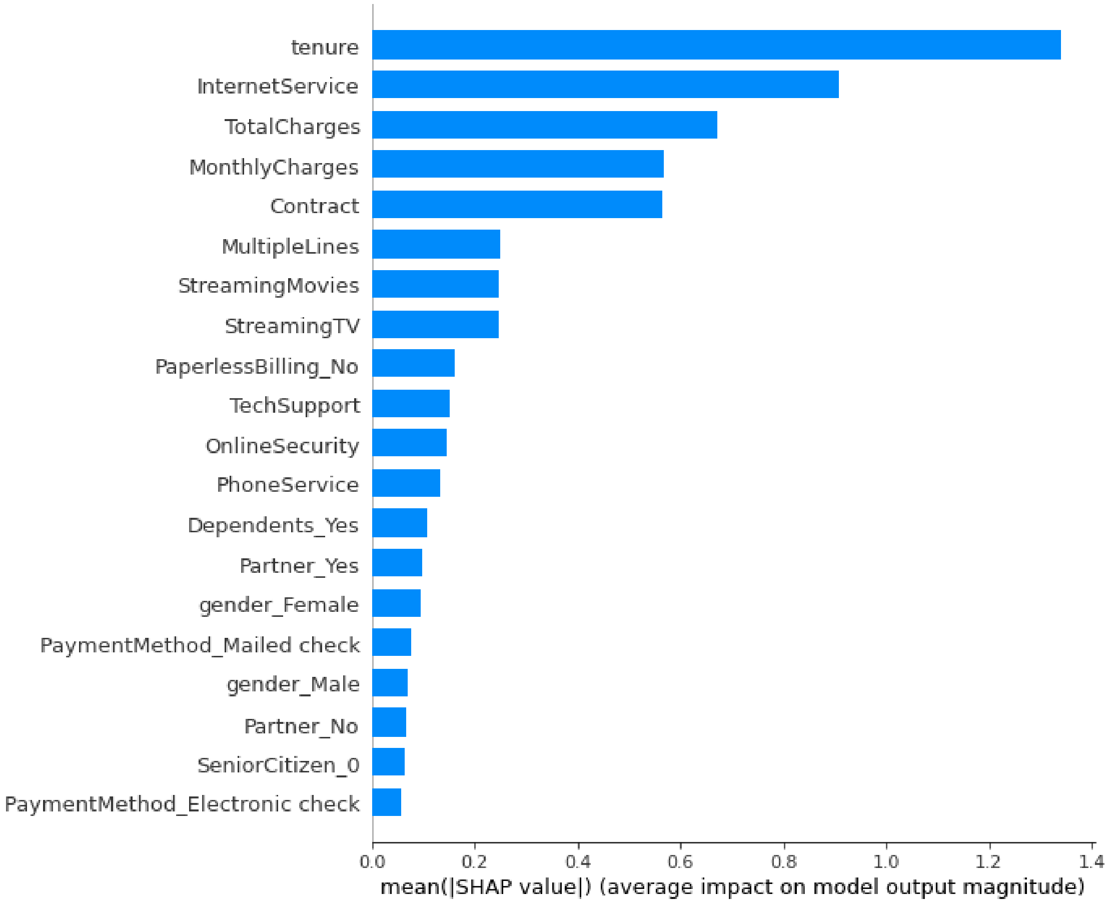
```
import shap
shap.initjs()
```

```
<IPython.core.display.HTML object>
```

[905]:

```
# Use the 10th model at random state 22*9 to calculate shap values

y_test = all_Y_test_v1[-1]
x_test = pd.DataFrame(all_X_test_v1[-1],columns = feature_names)
model = Logl2_models_v1[-1]
```

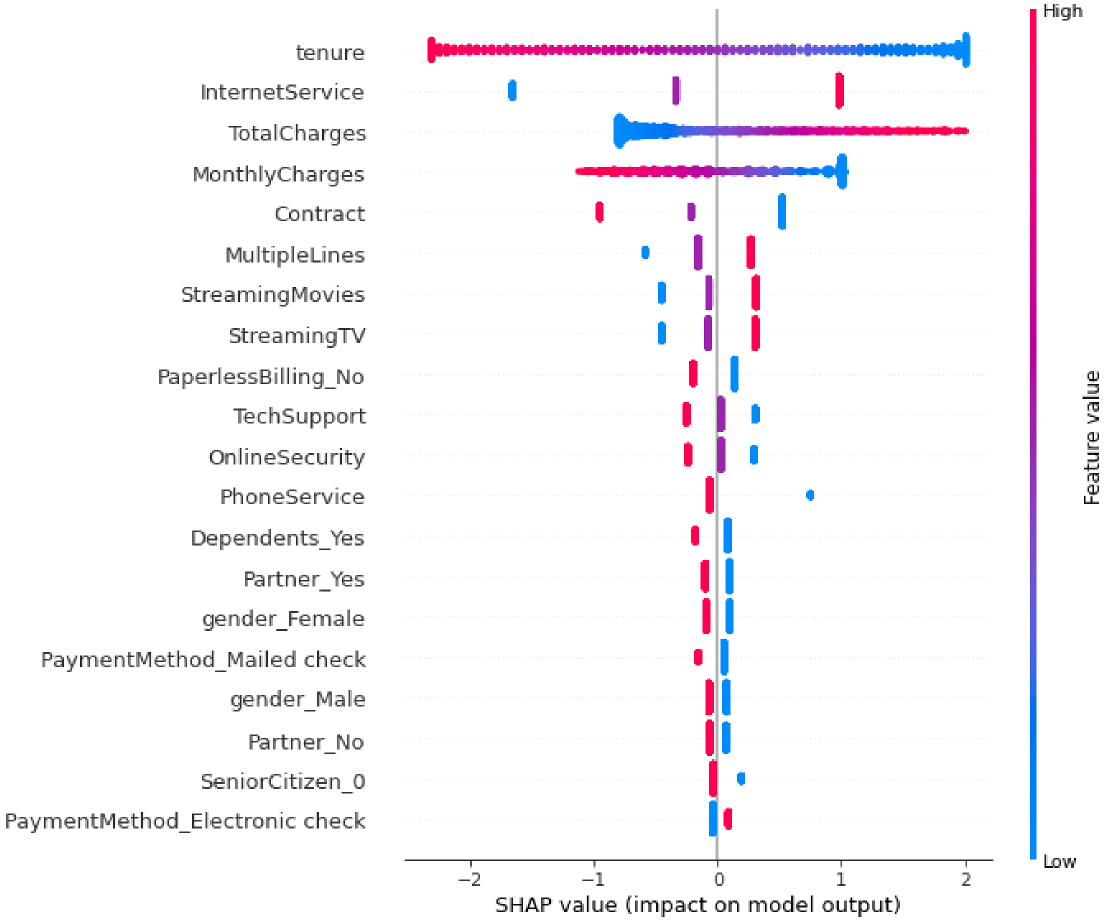
```
# Calculate Shap values for linear model
masker = shap.maskers.Independent(data=x_test, max_samples=1000)
explainer = shap.Explainer(model, masker=masker, feature_names= feature_names,␣
 ↪algorithm="linear")
shap_values = explainer.shap_values(x_test)

# Global Feature Importance based on Shap Values


shap.summary_plot(shap_values, x_test, plot_type="bar" ,feature_names=␣
 ↪feature_names, show=False)
plt.rcParams.update({'font.size': 13})
plt.savefig(dirct +'/figures/Shap_Global_Feature_Importance.png',dpi=300)
```



4.0.3 4.2 Local Feature Importance

[571]:
```
# Taking the last model and text data set to look into local feature importance␣
↪using Shap
```

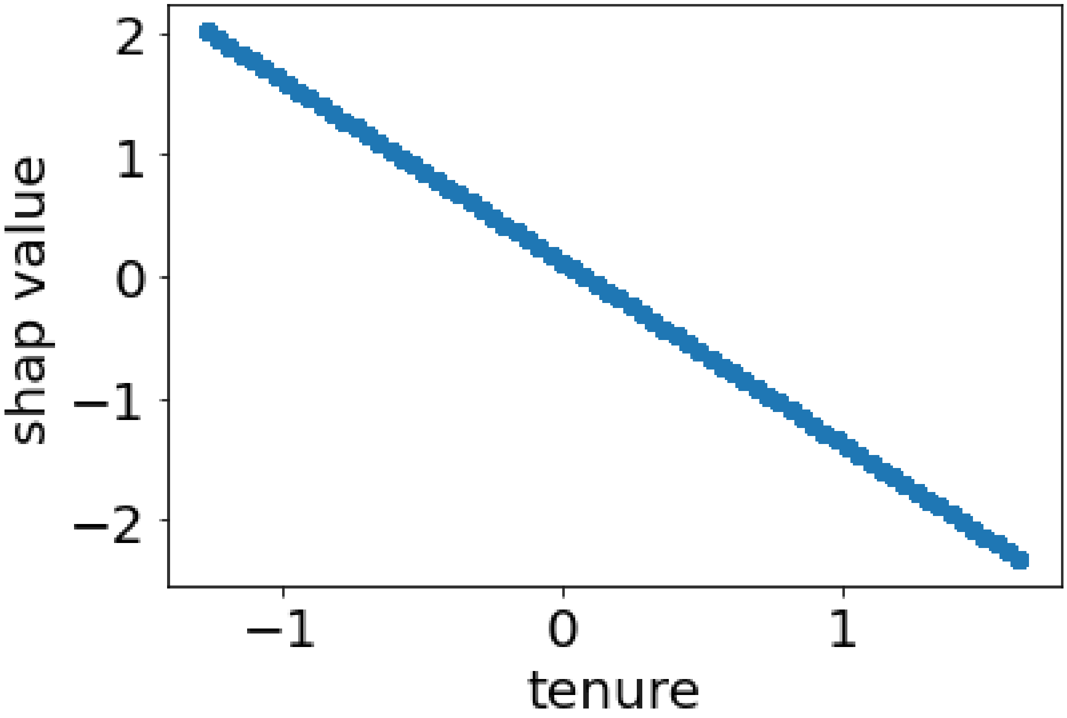
[794]:
```
shap.summary_plot(shap_values, x_test)
```



This figure combines the feature importance and its coefficient values. Features are ordered based on their importances. For example, higher total charges make customers more likely to leave.

[854]:
```
# Individual Feature: tenure

X_test_transformed = pd.DataFrame(x_test,columns = feature_names)
x_plot = X_test_transformed['tenure']

indx = feature_names.index('tenure')
indx

y_plot = shap_values[:,indx]
```

```
y_plot

matplotlib.rcParams.update({'font.size': 20})
plt.scatter(X_test_transformed['tenure'],shap_values[:,indx])
plt.ylabel('shap value')
plt.xlabel('tenure')
plt.show()
```

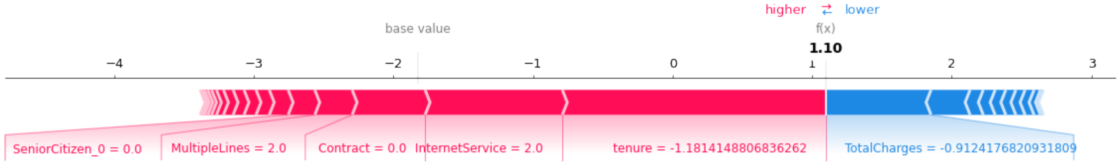


4.3.1 Local Feature Importance for Individual Datapoint

[915]:
```
## Looking at Costumer No.7

y_pred = model.predict(x_test)

print(x_test.iloc[7])
print(shap_values[7,:])
print('Predicted y:', y_pred[7])
print('True y: ', y_test.iloc[7])


shap.force_plot(explainer.expected_value,shap_values[7,:],X_test_transformed.
 ↪iloc[7],show = False, matplotlib = True, figsize=(20,3))
```

```
plt.savefig(dirct +'/figures/Feature_Contribution_Customer7.
 ↪png',dpi=300,bbox_inches = 'tight')
```

```
tenure                                      -1.181415
MonthlyCharges                               0.490783
TotalCharges                                -0.912418
gender_Female                                1.000000
gender_Male                                  0.000000
SeniorCitizen_0                              0.000000
SeniorCitizen_1                              1.000000
Partner_No                                   1.000000
Partner_Yes                                  0.000000
Dependents_No                                1.000000
Dependents_Yes                               0.000000
PaperlessBilling_No                          0.000000
PaperlessBilling_Yes                         1.000000
PaymentMethod_Bank transfer (automatic)      0.000000
PaymentMethod_Credit card (automatic)        0.000000
PaymentMethod_Electronic check               1.000000
PaymentMethod_Mailed check                   0.000000
PhoneService                                 1.000000
MultipleLines                                2.000000
InternetService                              2.000000
OnlineSecurity                               1.000000
OnlineBackup                                 2.000000
DeviceProtection                             1.000000
TechSupport                                  1.000000
StreamingTV                                  1.000000
StreamingMovies                              1.000000
Contract                                     0.000000
Name: 7, dtype: float64
[ 1.88712964 -0.27970072 -0.75400132 -0.09430171  0.06903318  0.1907135
 -0.08514097 -0.06409543  0.09461988 -0.02098952  0.07800132  0.13539609
  0.00208842  0.0188313   0.03479945  0.08634609  0.05023684 -0.06622797
  0.26751503  0.98311442  0.02707661 -0.04589792  0.00775927  0.02428314
 -0.07618488 -0.07185158  0.52258236]
Predicted y: 1
True y:   1
```

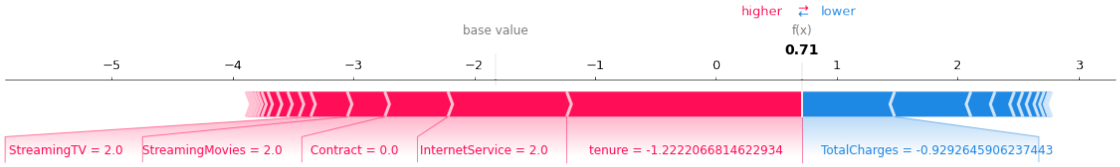


[916]:

```
# Looking at Costumer No.89

print(X_test_transformed.iloc[89])
print(shap_values[89,:])
print('Predicted y:', y_pred[89])
print('True y: ', y_test.iloc[89])

shap.force_plot(explainer.expected_value,shap_values[89],X_test_transformed.
 ↪iloc[89], show=False, matplotlib=True, figsize=(20,3))
plt.savefig(dirct +'/figures/Feature_Contribution_Customer89.png',dpi=300,␣
 ↪bbox_inches = 'tight')
```

```
tenure                                    -1.222207
MonthlyCharges                             1.034613
TotalCharges                              -0.929265
gender_Female                              0.000000
gender_Male                                1.000000
SeniorCitizen_0                            1.000000
SeniorCitizen_1                            0.000000
Partner_No                                 1.000000
Partner_Yes                                0.000000
Dependents_No                              1.000000
Dependents_Yes                             0.000000
PaperlessBilling_No                        1.000000
PaperlessBilling_Yes                       0.000000
PaymentMethod_Bank transfer (automatic)    0.000000
PaymentMethod_Credit card (automatic)      0.000000
PaymentMethod_Electronic check             1.000000
PaymentMethod_Mailed check                 0.000000
PhoneService                               1.000000
MultipleLines                              1.000000
InternetService                            2.000000
OnlineSecurity                             1.000000
OnlineBackup                               2.000000
DeviceProtection                           1.000000
TechSupport                                1.000000
StreamingTV                                2.000000
StreamingMovies                            2.000000
Contract                                   0.000000
Name: 89, dtype: float64
[ 1.94805589 -0.63072667 -0.76657849  0.09620679 -0.07042779 -0.0374127
  0.0167023  -0.06409543  0.09461988 -0.02098952  0.07800132 -0.19974274
 -0.00308094  0.0188313   0.03479945  0.08634609  0.05023684 -0.06622797
 -0.16050902  0.98311442  0.02707661 -0.04589792  0.00775927  0.02428314
  0.30284439  0.31033769  0.52258236]
Predicted y: 1
```

```
True y:   1
```

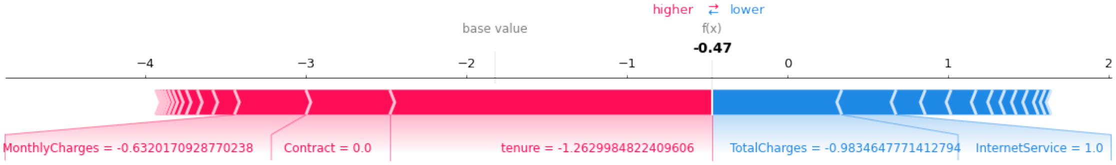


[917]:

```
# Looking at Costumer No.33


print(X_test_transformed.iloc[33])
print(shap_values[33,:])
print('Predicted y:', y_pred[33])
print('True y: ', y_test.iloc[33])


fig = plt.gcf()
shap.force_plot(explainer.expected_value,shap_values[33],X_test_transformed.
 ↪iloc[33], show=False, matplotlib=True,figsize=(20,3))
plt.savefig(dirct +'/figures/Feature_Contribution_Customer33.
 ↪png',dpi=300,bbox_inches = 'tight')
```

```
tenure                                     -1.262998
MonthlyCharges                             -0.632017
TotalCharges                               -0.983465
gender_Female                               0.000000
gender_Male                                 1.000000
SeniorCitizen_0                             1.000000
SeniorCitizen_1                             0.000000
Partner_No                                  0.000000
Partner_Yes                                 1.000000
Dependents_No                               0.000000
Dependents_Yes                              1.000000
PaperlessBilling_No                         0.000000
PaperlessBilling_Yes                        1.000000
PaymentMethod_Bank transfer (automatic)     0.000000
PaymentMethod_Credit card (automatic)       0.000000
PaymentMethod_Electronic check              0.000000
PaymentMethod_Mailed check                  1.000000
PhoneService                                1.000000
MultipleLines                               1.000000
InternetService                             1.000000
OnlineSecurity                              1.000000
```

```
OnlineBackup                              1.000000
DeviceProtection                          1.000000
TechSupport                               1.000000
StreamingTV                               1.000000
StreamingMovies                           1.000000
Contract                                  0.000000
Name: 33, dtype: float64
[ 2.00898214  0.44503286 -0.80704198  0.09620679 -0.07042779 -0.0374127
  0.0167023   0.06915911 -0.10209505  0.04897555 -0.18200307  0.13539609
  0.00208842  0.0188313   0.03479945 -0.04252867 -0.15649914 -0.06622797
 -0.16050902 -0.33827593  0.02707661  0.00861267  0.00775927  0.02428314
 -0.07618488 -0.07185158  0.52258236]
Predicted y: 0
True y:  1

<Figure size 432x288 with 0 Axes>
```


higher ⇄ lower
base value
f(x)
MonthlyCharges = -0.6320170928770238
Contract = 0.0
tenure = -1.2629984822409606
TotalCharges = -0.9834647771412794
InternetService = 1.0