mvp_severity_socioeconomic

May 2, 2020

[46]:
```
# store start time to get execution time of entire script
import time
start_time = time.time()
```

[47]:
```
# helper functions for displaying table data

import numpy as np
from IPython.display import display_html

# n is the number of columns to display data in
def display_side_by_side(series_obj, n):
    df = pd.DataFrame(series_obj)
    partition = int(round(len(df) / n))
    lower_bound = 0
    upper_bound = partition
    args = []
    for i in range(n):
        args.append(df[lower_bound:upper_bound])
        lower_bound += partition
        upper_bound += partition
    helper(args)

def helper(args):
    html_str=''
    for df in args:
        html_str+=df.to_html()
    display_html(html_str.replace('table','table style="display:
 ↪inline"'),raw=True)
```

[48]:
```
# helper function for plotting out ground truth curves

import matplotlib.pyplot as plt
plt.rcParams["font.weight"] = "bold"

def get_ground_truth(data):
    relapsed = data[data.Illicit_Cens5 == 1]
    counts = relapsed['Illicit_Days5'].value_counts()
```

```
    counts = counts.to_dict()
    temp = [len(data)] * 365
    labels = list(range(365))
    for i in range(365):
        labels[i] += 1
    total = 0
    errors = []
    for i in range(365):
        try:
            temp[i] = temp[i] - counts[i+1] - total
            total = total + counts[i+1]
        except KeyError:
            errors.append(i)

    for ele in sorted(errors, reverse = False):
        if ele != 0:
            temp[ele] = temp[ele-1]
        else:
             temp[0] = len(data)
    temp = [x / len(data) for x in temp]
    return labels, temp
```

[49]:
```
from sklearn.model_selection import cross_validate
from sksurv.ensemble import GradientBoostingSurvivalAnalysis
from sksurv.ensemble import RandomSurvivalForest
from sksurv.linear_model import CoxnetSurvivalAnalysis

def run_models(X, y, label):
    gbsa = GradientBoostingSurvivalAnalysis()
    scores = cross_validate(gbsa, X, y, cv=5)
    gbsa_score = scores['test_score'].mean()
    print('RF Boosted score:', gbsa_score)

    gbsa = GradientBoostingSurvivalAnalysis()
    gbsa.fit(X, y)

    # selected via forward/backward feature selection
    """temp =␣
↪X[["%_U18g","%_female_householdg","%_povertyg","B2a_0g","SESg_0_cd","cjsig_0_cd","dssg_0_cd
            ␣
↪"primsev_cd_4","primsev_cd_5","primsev_cd_6","prsatx_cd","r4ag_0_cd","srig_0_cd","unemplmt_
            ␣
↪"%_public_assistanceg","%_unemployedg","CWSg_0_cd","adhdg_0_cd","cdsg_0_cd","epsg_0_cd","fe
             "lrig_0_cd","nonwhite_cd","primsev_cd_2","primsev_cd_3"]]"""
    rsf = RandomSurvivalForest()
    """scores = cross_validate(rsf, temp, y, cv=5)"""
    scores = cross_validate(rsf, X, y, cv=5)
```

```
    rsf_score = scores['test_score'].mean()
    print('RF score:', rsf_score)

    rsf = RandomSurvivalForest()
    """rsf.fit(temp, y)"""
    rsf.fit(X, y)

    # selected via forward/backward feature selection
    """temp =␣
↪X[["%_U18g","%_female_householdg","%_unemployedg","SESg_0_cd","adhdg_0_cd","cdsg_0_cd","cjs
                ␣
↪"dssg_0_cd","nonwhite_cd","primsev_cd_3","primsev_cd_4","primsev_cd_6","prsatx_cd","r4ag_0_
                ␣
↪"srig_0_cd","epsg_0_cd","gvsg_cd","primsev_cd_1","primsev_cd_2","unemplmt_cd"]]"""
    # l1_ratio = 1 adjusts model to implement LASSO method for penalties
    rcr = CoxnetSurvivalAnalysis(l1_ratio=1)
    """scores = cross_validate(rcr, temp, y, cv=5)"""
    scores = cross_validate(rcr, X, y, cv=5)
    rcr_score = scores['test_score'].mean()
    print('Lasso score:', rcr_score)

    # fit_baseline_model = True allows us to create survival/hazard plots after␣
↪model is fit
    rcr = CoxnetSurvivalAnalysis(fit_baseline_model=True, l1_ratio=1)
    """rcr.fit(temp, y)"""
    rcr.fit(X, y)

    # concordance index
    scores = {'Model': ['Random Forest Boosted','Random␣
↪Forest','Lasso','Dataset Size'],
                label: [gbsa_score,rsf_score,rcr_score,X.shape[0]]}

    concordance = pd.DataFrame(data=scores)

    # return scores and models
    return concordance, gbsa, rsf, rcr
```

[50]:
```
def get_survival_graph(rsf, rcr, X, Y, label, filename):
    """temp =␣
↪X[["%_U18g","%_female_householdg","%_povertyg","B2a_0g","SESg_0_cd","cjsig_0_cd","dssg_0_cd
                ␣
↪"primsev_cd_4","primsev_cd_5","primsev_cd_6","prsatx_cd","r4ag_0_cd","srig_0_cd","unemplmt_
                ␣
↪"%_public_assistanceg","%_unemployedg","CWSg_0_cd","adhdg_0_cd","cdsg_0_cd","epsg_0_cd","fe
                 "lrig_0_cd","nonwhite_cd","primsev_cd_2","primsev_cd_3"]]
    pred_surv_rsf = rsf.predict_survival_function(temp)"""
```

```
    pred_surv_rsf = rsf.predict_survival_function(X)
    """temp =␣
↪X[["%_U18g","%_female_householdg","%_unemployedg","SESg_0_cd","adhdg_0_cd","cdsg_0_cd","cjs
                ␣
↪"dssg_0_cd","nonwhite_cd","primsev_cd_3","primsev_cd_4","primsev_cd_6","prsatx_cd","r4ag_0_
                ␣
↪"srig_0_cd","epsg_0_cd","gvsg_cd","primsev_cd_1","primsev_cd_2","unemplmt_cd"]]
    pred_surv_rcr = rcr.predict_survival_function(temp)"""
    pred_surv_rcr = rcr.predict_survival_function(X)

    # display survival plot
    plt.suptitle(label)
    plt.plot(np.mean([person for person in pred_surv_rsf], axis=0), label='RF')
    plt.plot(np.mean([person.y for person in pred_surv_rcr], axis=0),␣
↪label='Lasso')
    labels, temp = get_ground_truth(Y)
    plt.plot(labels, temp, label='Ground Truth')
    plt.legend()
    plt.xlim(0, 365)
    plt.xticks(np.arange(0, 365, step=50))
    plt.yticks(np.arange(0, 1.1, step=0.1))
    plt.savefig(filename)

    plt.show()
```

[51]:
```
def get_feature_importance(features, gbsa, rcr, label):
    # feature importances from Boosted Random Forest
    feature_importance_rf = pd.DataFrame({'Feature':features, label:gbsa.
↪feature_importances_,})
    feature_importance_rf.sort_values(by=[label], ascending=False, inplace=True)
    feature_importance_rf = feature_importance_rf.nlargest(10,[label]) # keep␣
↪top 10 features
    feature_importance_rf = feature_importance_rf[feature_importance_rf[label] !
↪= 0]

    # feature importances from Lasso
    """temp =␣
↪["%_U18g","%_female_householdg","%_unemployedg","SESg_0_cd","adhdg_0_cd","cdsg_0_cd","cjsig
                ␣
↪"dssg_0_cd","nonwhite_cd","primsev_cd_3","primsev_cd_4","primsev_cd_6","prsatx_cd","r4ag_0_
                ␣
↪"srig_0_cd","epsg_0_cd","gvsg_cd","primsev_cd_1","primsev_cd_2","unemplmt_cd"]"""
    feature_importance_lasso = pd.DataFrame({'Feature':features, #temp,
                                             label:np.average(rcr.coef_,␣
↪weights=rcr.alphas_, axis = 1),})
```

```
    feature_importance_lasso[label + '_abs'] = np.
↪absolute(feature_importance_lasso[label])
    feature_importance_lasso = feature_importance_lasso.nlargest(10,[label +␣
↪'_abs']) # keep top 10 features
    feature_importance_lasso =␣
↪feature_importance_lasso[feature_importance_lasso[label] != 0]

    return feature_importance_rf, feature_importance_lasso
```

Survival Analysis by Severity

[52]:
```
import pandas as pd
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
import csv

df = pd.read_csv('data/data_superset.csv')
df.drop(columns=['SDS1_0'], inplace=True)
df.head()
```

[52]:

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	ID	State	City	\
0	0	0	1	857	OH	Cleveland	
1	1	1	2	929	OH	Cleveland	
2	2	2	3	951	OH	Cleveland	
3	3	3	4	1032	OH	Cleveland	
4	4	4	5	1039	OH	Cleveland	

	agyaddr	xobsyr_0	Illicit_Days5	Illicit_Cens5	\
0	1276 West Third St. #400	2005	365	0	
1	1276 West Third St. #400	2006	354	0	
2	1276 West Third St. #400	2006	365	0	
3	1276 West Third St. #400	2006	365	0	
4	1276 West Third St. #400	2004	365	0	

	female_cd	nonwhite_cd	unemplmt_cd	prsatx_cd	gvsg_cd	CWSg_0_cd	\
0	0	0	0	0	2	0	
1	0	0	0	0	1	0	
2	0	0	0	0	0	0	
3	0	0	0	0	2	0	
4	0	0	0	0	0	0	

	srprobg_cd	dssg_0_cd	epsg_0_cd	adhdg_0_cd	cdsg_0_cd	cjsig_0_cd	\
0	0	1	1	1	1	0	
1	1	0	1	0	1	1	
2	1	0	0	0	0	1	
3	1	1	1	1	1	0	
4	0	1	0	0	1	1	

```
   lrig_0_cd  srig_0_cd  SESg_0_cd  r4ag_0_cd  nonillicit_flag  primsev_cd_1  \
0          1          2          0          1                1             1
1          0          1          0          2                1             1
2          0          1          0          2                1             0
3          2          1          0          2                1             0
4          2          1          0          2                0             0

   primsev_cd_2  primsev_cd_3  primsev_cd_4  primsev_cd_5  primsev_cd_6  \
0             0             0             0             0             0
1             0             0             0             0             0
2             0             1             0             0             0
3             0             1             0             0             0
4             1             0             0             0             0

   B2a_0g  SUDSy_0                                  Address        lat  \
0       0        0  1276 West Third St. #400, Cleveland, OH  41.501028
1       0        3  1276 West Third St. #400, Cleveland, OH  41.501028
2       0        2  1276 West Third St. #400, Cleveland, OH  41.501028
3       0        2  1276 West Third St. #400, Cleveland, OH  41.501028
4       0        0  1276 West Third St. #400, Cleveland, OH  41.501028

         lng state_name  Geo_FIPS    block_FIPS  murder_numg  gran  pop_deng  \
0 -81.697772       Ohio   39035.0  3.903511e+14          0.0   NaN       NaN
1 -81.697772       Ohio   39035.0  3.903511e+14          0.0   0.0       0.0
2 -81.697772       Ohio   39035.0  3.903511e+14          0.0   0.0       0.0
3 -81.697772       Ohio   39035.0  3.903511e+14          0.0   0.0       0.0
4 -81.697772       Ohio   39035.0  3.903511e+14          0.0   NaN       NaN

   %_dropoutg  %_unemployedg  %_public_assistanceg  %_povertyg
0         NaN            NaN                   NaN         NaN
1         1.0            1.0                   1.0         0.0
2         1.0            1.0                   1.0         0.0
3         1.0            1.0                   1.0         0.0
4         NaN            NaN                   NaN         NaN
```

[53]:
```
# drop unnecessary columns
cols_to_drop = ['Address','lat','lng','xobsyr_0','Unnamed: 0','Unnamed: 0.
 ↪1','Unnamed: 0.1.1',
                'ID','State','City','agyaddr','state_name','gran','srprobg_cd']

# uncomment for stratified
#cols_to_drop += ['county_FIPS_x','county_FIPS_y','block_FIPS_x','block_FIPS_y']

# uncomment for regular
cols_to_drop += ['Geo_FIPS','block_FIPS']
```

```
# uncomment to get CONTROL statistics
#cols_to_drop = cols_to_drop +␣
 ↪['pop_den','%_dropout','%_unemployedg','%_public_assistanceg','%_povertyg','murder_numg']

df.drop(columns=cols_to_drop, inplace=True)
df.dropna(inplace=True)
df = df.astype(int)
df.shape
```

[53]: (9085, 32)

[54]:
```
# df = df[df.nonillicit_flag == 0] # subset to only the illicit cases
df.drop(columns=['nonillicit_flag'], inplace=True) # if not used to subset,␣
 ↪remove feature since its redundant
```

[55]:
```
df.shape
```

[55]: (9085, 31)

[56]:
```
df.head()
```

[56]:

	Illicit_Days5	Illicit_Cens5	female_cd	nonwhite_cd	unemplmt_cd	\
1	354	0	0	0	0	
2	365	0	0	0	0	
3	365	0	0	0	0	
18	365	0	0	0	0	
21	5	1	0	0	0	

	prsatx_cd	gvsg_cd	CWSg_0_cd	dssg_0_cd	epsg_0_cd	adhdg_0_cd	\
1	0	1	0	0	1	0	
2	0	0	0	0	0	0	
3	0	2	0	1	1	1	
18	0	0	0	0	1	1	
21	1	2	1	2	2	1	

	cdsg_0_cd	cjsig_0_cd	lrig_0_cd	srig_0_cd	SESg_0_cd	r4ag_0_cd	\
1	1	1	0	1	0	2	
2	0	1	0	1	0	2	
3	1	0	2	1	0	2	
18	0	0	2	2	0	0	
21	2	1	1	2	2	2	

	primsev_cd_1	primsev_cd_2	primsev_cd_3	primsev_cd_4	primsev_cd_5	\
1	1	0	0	0	0	
2	0	0	1	0	0	
3	0	0	1	0	0	
18	0	0	1	0	0	

```
21              0              0              1              0              0

    primsev_cd_6  B2a_0g  SUDSy_0  murder_numg  pop_deng  %_dropoutg  \
1              0       0        3            0         0           1
2              0       0        2            0         0           1
3              0       0        2            0         0           1
18             0       0        2            0         0           1
21             0       0       11            0         0           1

    %_unemployedg  %_public_assistanceg  %_povertyg
1               1                     1           0
2               1                     1           0
3               1                     1           0
18              0                     0           0
21              0                     0           0
```

Full Population Survival Analysis

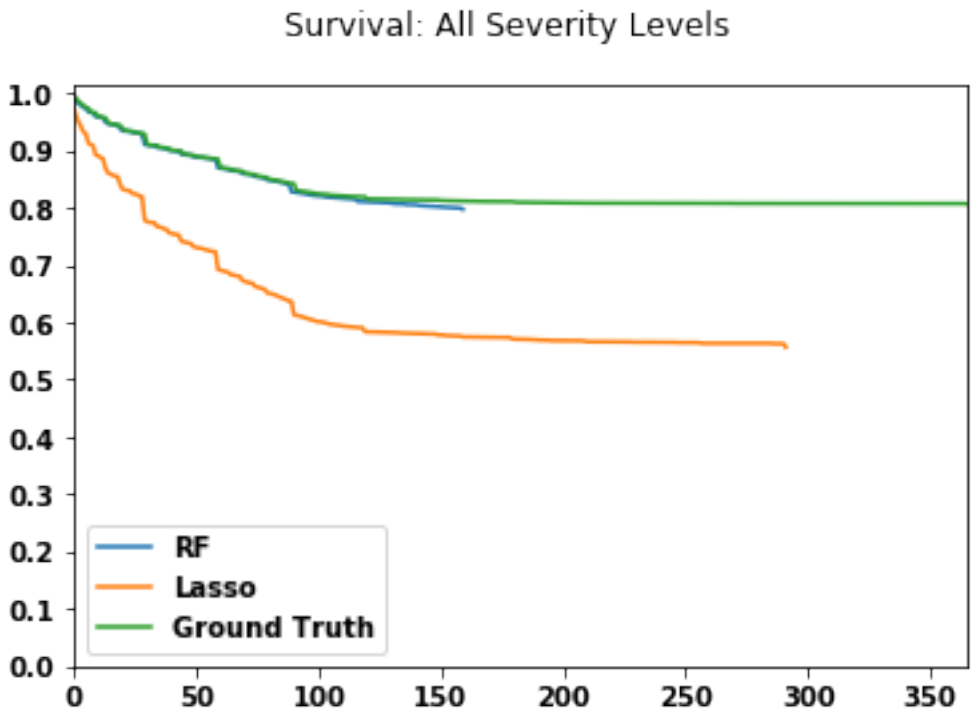
[57]:
```
from sklearn.model_selection import train_test_split
from sksurv.util import Surv

predictor_var = 'Illicit_Days5'
censoring_var = 'Illicit_Cens5'

X = df.copy()
Y = X[[censoring_var, predictor_var]]
X.drop(columns=[censoring_var, predictor_var], inplace=True)
y = Surv.from_arrays(Y[censoring_var], Y[predictor_var]) # structured array to␣
↪ensure correct censoring

print(X.shape, y.shape)
```

```
(9085, 29) (9085,)
```

[58]:
```
%%time
full_concordance, gbsa, rsf, rcr = run_models(X, y, 'ALL')
```

```
RF Boosted score: 0.6715182758422004
RF score: 0.6704720332717683
Lasso score: 0.674120464660447
CPU times: user 8min 24s, sys: 4.7 s, total: 8min 29s
Wall time: 8min 41s
```

[59]:
```
get_survival_graph(rsf, rcr, X, Y, 'Survival: All Severity Levels','graphs/
↪survival_all.png')
```



Subclinical Severity Survival Analysis

[60]:
```
X = df[df.SUDSy_0 < 2]
Y = X[[censoring_var, predictor_var]]
X.drop(columns=[censoring_var, predictor_var, 'SUDSy_0'], inplace=True)

y = Surv.from_arrays(Y[censoring_var], Y[predictor_var]) # structured array to␣
 ↪ensure correct censoring

print(X.shape, y.shape)
```

```
(3029, 28) (3029,)
```

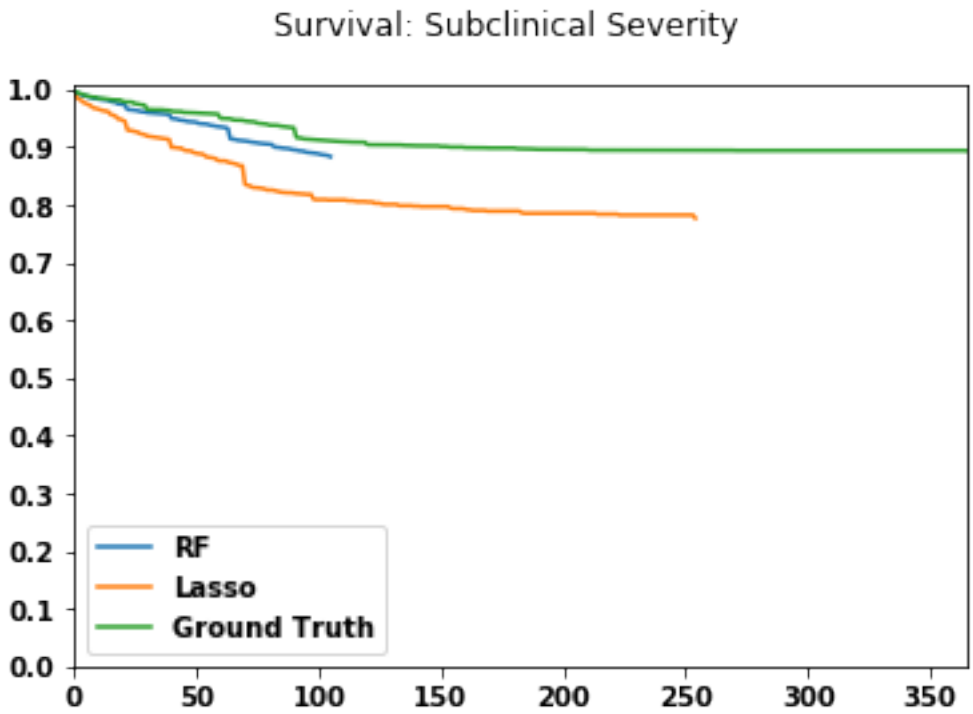
```
//anaconda3/lib/python3.7/site-packages/pandas/core/frame.py:4097:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  errors=errors,
```

[61]:
```
%%time
subclinical_concordance, gbsa, rsf, rcr = run_models(X, y, 'SUB')
```

```
RF Boosted score: 0.6245246599444958
```

```
RF score: 0.6261989581327161
Lasso score: 0.6490158664689396
CPU times: user 54.6 s, sys: 431 ms, total: 55.1 s
Wall time: 54.9 s
```

[62]:
```
get_survival_graph(rsf, rcr, X, Y, 'Survival: Subclinical Severity', 'graphs/
↪survival_subclinical.png')
```



[63]:
```
subclinical_feature_importance_rf, subclinical_feature_importance_lasso = \
                                        get_feature_importance(X.columns, gbsa,␣
↪rcr, 'SUB')
```

Mild/Moderate Severity Survival Analysis

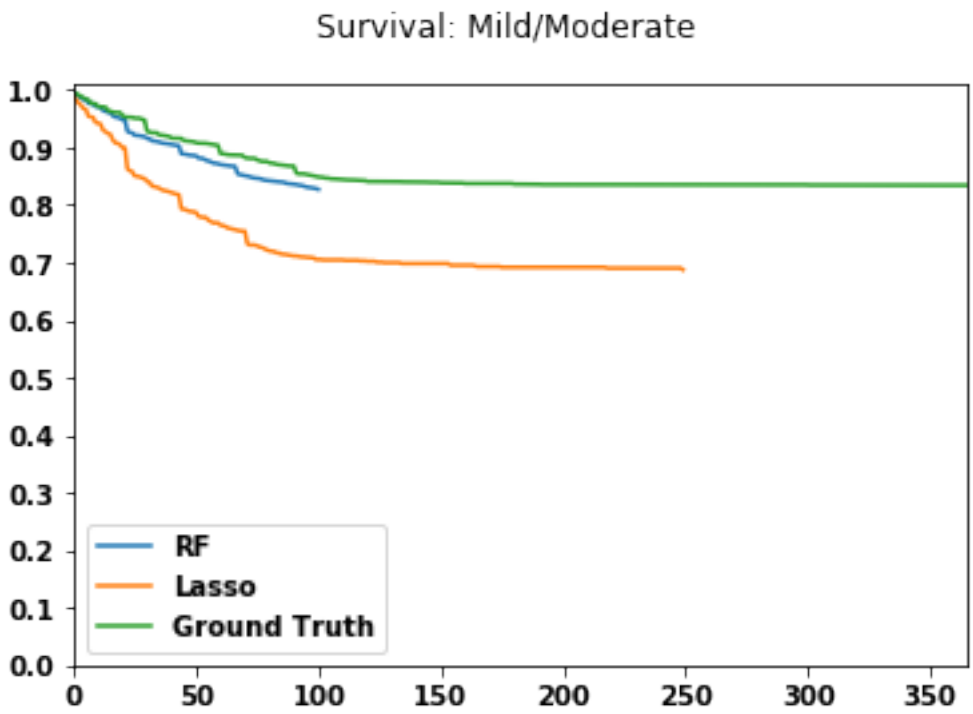
[64]:
```
X = df[df.SUDSy_0 >= 2]
X = X[X.SUDSy_0 <= 5]
Y = X[[censoring_var, predictor_var]]
X.drop(columns=[censoring_var, predictor_var, 'SUDSy_0'], inplace=True)

y = Surv.from_arrays(Y[censoring_var], Y[predictor_var]) # structured array to␣
↪ensure correct censoring

print(X.shape, y.shape)
```

```
(2565, 28) (2565,)
```

[65]:
```
%%time
mild_concordance, gbsa, rsf, rcr = run_models(X, y, 'MILD')
```

```
RF Boosted score: 0.552435439158523
RF score: 0.5601322809331213
Lasso score: 0.560069039243234
CPU times: user 41.4 s, sys: 315 ms, total: 41.7 s
Wall time: 41.4 s
```

[66]:
```
get_survival_graph(rsf, rcr, X, Y, 'Survival: Mild/Moderate', 'graphs/
↪survival_mild.png')
```



[67]:
```
mild_feature_importance_rf, mild_feature_importance_lasso =␣
↪get_feature_importance(X.columns, gbsa, rcr, 'MILD')
```

Severe Severity Survival Analysis

[68]:
```
X = df[df.SUDSy_0 > 5]
Y = X[[censoring_var, predictor_var]]
X.drop(columns=[censoring_var, predictor_var, 'SUDSy_0'], inplace=True)
```

```
y = Surv.from_arrays(Y[censoring_var], Y[predictor_var]) # structured array to␣
 ↪ensure correct censoring

print(X.shape, y.shape)
```

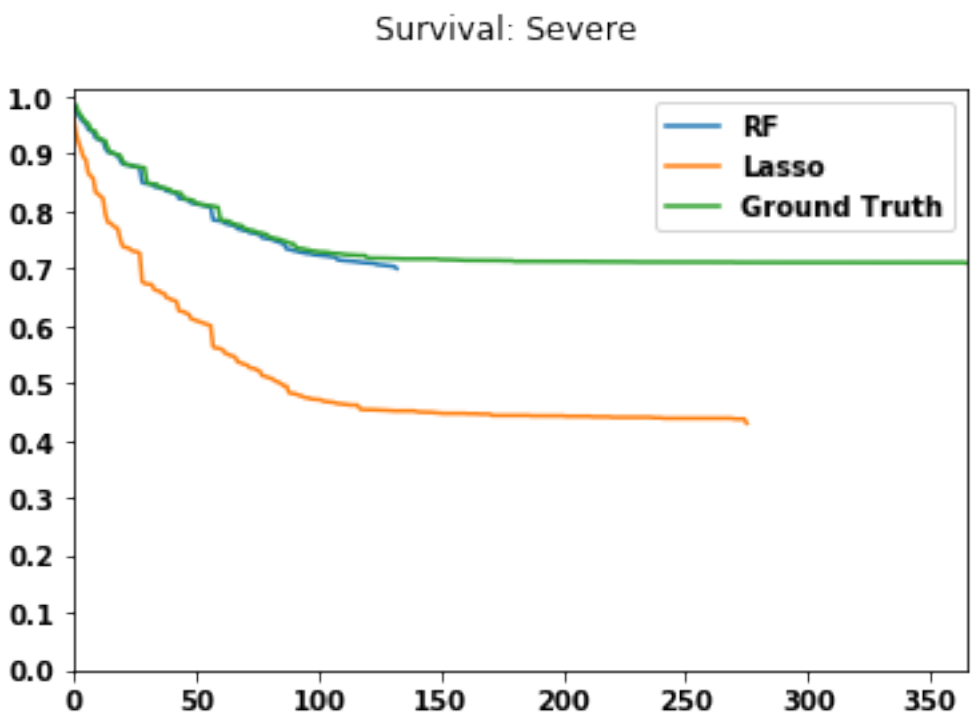
```
(3491, 28) (3491,)
```

```
//anaconda3/lib/python3.7/site-packages/pandas/core/frame.py:4097:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  errors=errors,
```

[69]:
```
%%time
severe_concordance, gbsa, rsf, rcr = run_models(X, y, 'SEVERE')
```

```
RF Boosted score: 0.6037516196914867
RF score: 0.597467726642787
Lasso score: 0.6042207532482173
CPU times: user 1min 22s, sys: 713 ms, total: 1min 22s
Wall time: 1min 22s
```

[70]:
```
get_survival_graph(rsf, rcr, X, Y, 'Survival: Severe', 'graphs/survival_severe.
 ↪png')
```



[71]:
```
severe_feature_importance_rf, severe_feature_importance_lasso =␣
↪get_feature_importance(X.columns, gbsa, rcr, 'SEVERE')
```

Overall Statistics

[72]:
```
overall_concordance = pd.concat([subclinical_concordance,␣
↪mild_concordance['MILD'], severe_concordance['SEVERE'],
                                  full_concordance['ALL']], axis=1)
pd.DataFrame(data=overall_concordance).round(4)
```

[72]:

	Model	SUB	MILD	SEVERE	ALL
0	Random Forest Boosted	0.6245	0.5524	0.6038	0.6715
1	Random Forest	0.6262	0.5601	0.5975	0.6705
2	Lasso	0.6490	0.5601	0.6042	0.6741
3	Dataset Size	3029.0000	2565.0000	3491.0000	9085.0000

[73]:
```
overall_feature_importance_lasso = pd.
↪merge(subclinical_feature_importance_lasso, \
                                          mild_feature_importance_lasso,␣
↪on='Feature', how='outer')
overall_feature_importance_lasso = pd.merge(overall_feature_importance_lasso, \
                                          severe_feature_importance_lasso,␣
↪on='Feature', how='outer')
overall_feature_importance_lasso.fillna(0, inplace=True)
display_side_by_side(overall_feature_importance_lasso, 2)
```

[74]:
```
hazards = overall_feature_importance_lasso[['SUB','MILD','SEVERE','Feature']]
hazards['exp(SUB)-1'] = np.exp(overall_feature_importance_lasso['SUB']) - 1
hazards['exp(MILD)-1'] = np.exp(overall_feature_importance_lasso['MILD']) - 1
hazards['exp(SEVERE)-1'] = np.exp(overall_feature_importance_lasso['SEVERE']) -␣
↪1

hazards.head()
```

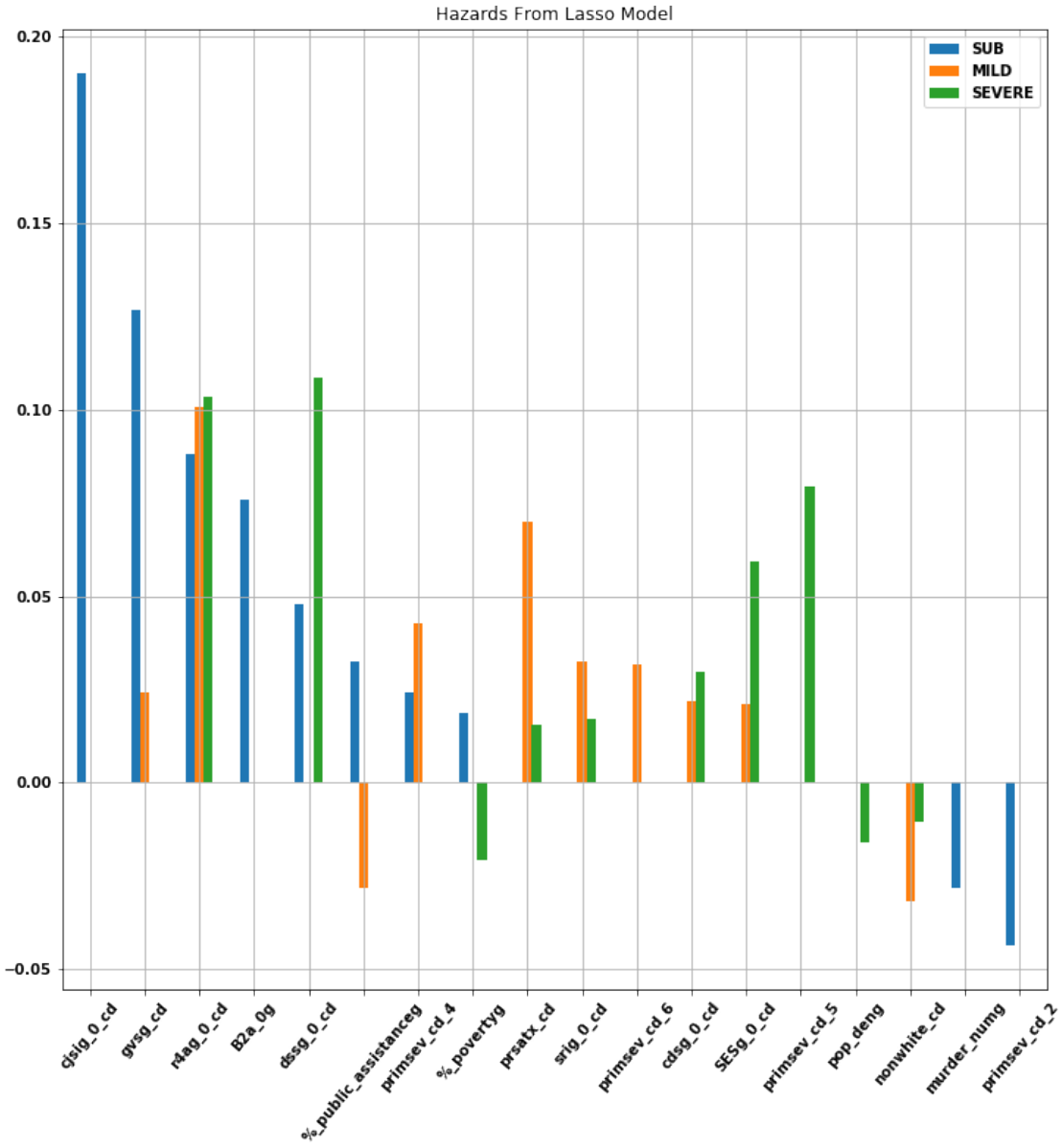
```
//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[74]:

	SUB	MILD	SEVERE	Feature	exp(SUB)-1	exp(MILD)-1	exp(SEVERE)-1
0	0.174109	0.000000	0.000000	cjsig_0_cd	0.190186	0.000000	0.000000
1	0.119116	0.024083	0.000000	gvsg_cd	0.126501	0.024375	0.000000
2	0.084463	0.096058	0.098269	r4ag_0_cd	0.088133	0.100823	0.103260
3	0.073101	0.000000	0.000000	B2a_0g	0.075839	0.000000	0.000000
4	0.046614	0.000000	0.102892	dssg_0_cd	0.047718	0.000000	0.108372

[75]:
```
haz_df = pd.DataFrame({'SUB': hazards['exp(SUB)-1'].tolist(),
                       'MILD': hazards['exp(MILD)-1'].tolist(),
                       'SEVERE': hazards['exp(SEVERE)-1'].tolist()},
                      index=hazards['Feature'].tolist())
haz_df.sort_values(by=['SUB','MILD','SEVERE'], ascending=False, inplace=True)
ax = haz_df.plot.bar(rot=50, figsize=(12, 12))
ax.grid()
ax.set_title('Hazards From Lasso Model')
fig = ax.get_figure()
```



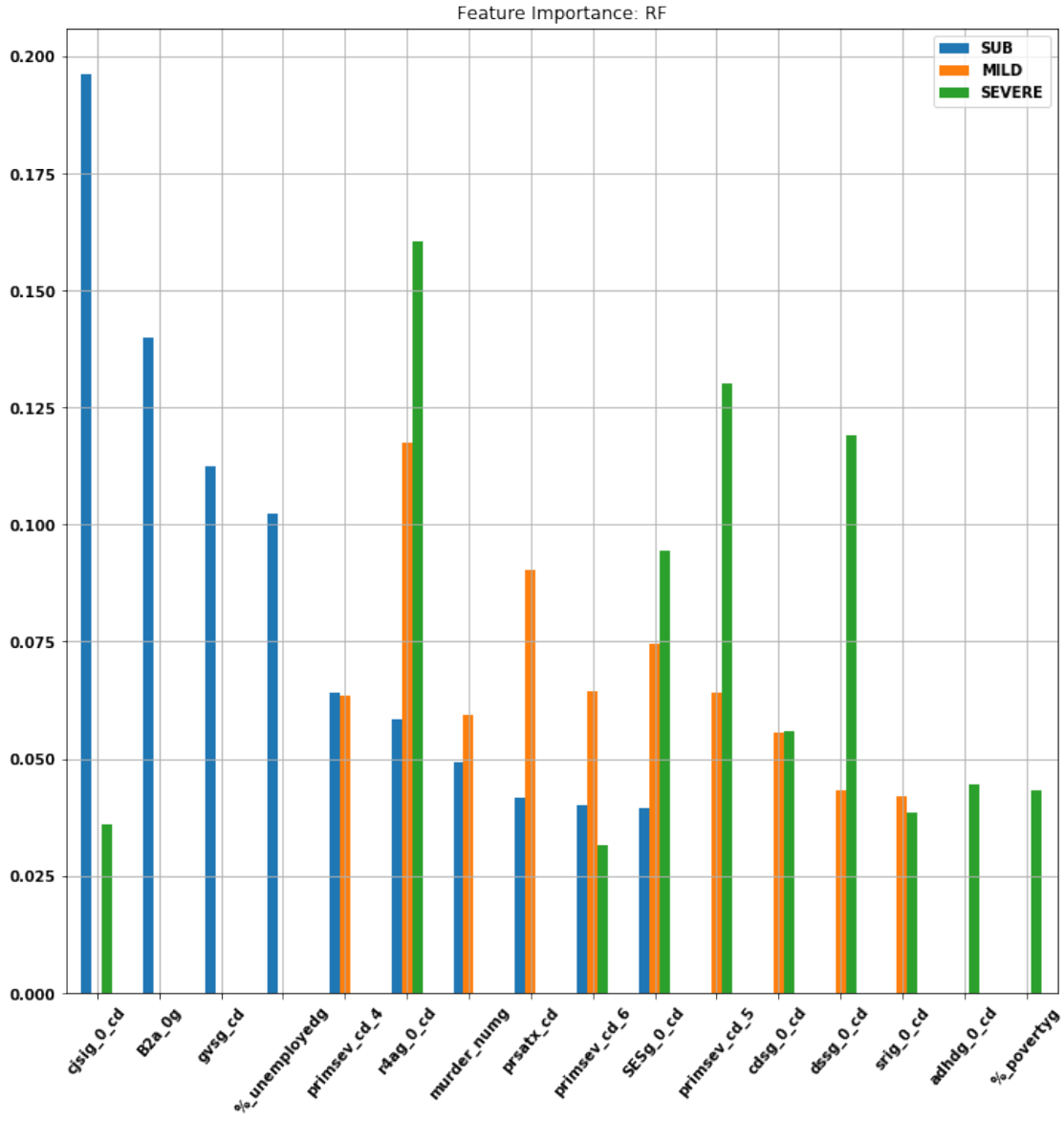
[76]:
```
# feature importance for lasso across all ages
"""df = pd.DataFrame({'SUB': overall_feature_importance_lasso['SUB'].tolist(),
                      'MILD': overall_feature_importance_lasso['MILD'].tolist(),
                      'SEVERE': overall_feature_importance_lasso['SEVERE'].
 ↪tolist()},
                     index=overall_feature_importance_lasso['Feature'].tolist())
df.sort_values(by=['SUB','MILD','SEVERE'], ascending=False, inplace=True)
ax = df.plot.bar(rot=50, figsize=(12, 12))
ax.grid()
ax.set_title('Feature Importance: Lasso')
```

```
fig = ax.get_figure()

fig.savefig('graphs/feature_importance_lasso.png', bbox_inches='tight')"""
```

[76]:
```
'plt.rcParams["font.weight"] = "bold"\n\ndf = pd.DataFrame({\'SUB\':
overall_feature_importance_lasso[\'SUB\'].tolist(),\n
\'MILD\': overall_feature_importance_lasso[\'MILD\'].tolist(),\n
\'SEVERE\': overall_feature_importance_lasso[\'SEVERE\'].tolist()},\n
index=overall_feature_importance_lasso[\'Feature\'].tolist())\ndf.sort_values(by
=[\'SUB\',\'MILD\',\'SEVERE\'], ascending=False, inplace=True)\nax =
df.plot.bar(rot=50, figsize=(12, 12))\nax.grid()\nax.set_title(\'Feature
Importance: Lasso\')\nfig = ax.get_figure()\n
\nfig.savefig(\'graphs/feature_importance_lasso.png\', bbox_inches=\'tight\')'
```

[77]:
```
overall_feature_importance_rf = pd.merge(subclinical_feature_importance_rf,␣
↪mild_feature_importance_rf, on='Feature', how='outer')
overall_feature_importance_rf = pd.merge(overall_feature_importance_rf,␣
↪severe_feature_importance_rf, on='Feature', how='outer')
overall_feature_importance_rf.fillna(0, inplace=True)
display_side_by_side(overall_feature_importance_rf, 4)
```

[78]:
```
# feature importance for rf across all ages
df = pd.DataFrame({'SUB': overall_feature_importance_rf['SUB'].tolist(),
                   'MILD': overall_feature_importance_rf['MILD'].tolist(),
                   'SEVERE': overall_feature_importance_rf['SEVERE'].tolist()},
                  index=overall_feature_importance_rf['Feature'].tolist())
df.sort_values(by=['SUB','MILD','SEVERE'], ascending=False, inplace=True)
ax = df.plot.bar(rot=50, figsize=(12, 12))
ax.grid()
ax.set_title('Feature Importance: RF')
fig = ax.get_figure()

fig.savefig('graphs/feature_importance_rf.png', bbox_inches='tight')
```



[79]:
```
# features in top 10 of both models across all ages
feature_importance_intersection = np.
 ↪intersect1d(overall_feature_importance_rf['Feature'],
                                                  ␣
 ↪overall_feature_importance_lasso['Feature'])
print('Common Features:', *list(feature_importance_intersection), sep =', ')
```

```
Common Features:, %_povertyg, B2a_0g, SESg_0_cd, cdsg_0_cd, cjsig_0_cd,
dssg_0_cd, gvsg_cd, murder_numg, primsev_cd_4, primsev_cd_5, primsev_cd_6,
prsatx_cd, r4ag_0_cd, srig_0_cd
```

[80]:
```
# print out total notebook execution time
total_seconds = int(time.time() - start_time)
minutes = total_seconds // 60
seconds = total_seconds % 60
print("--- " + str(minutes) + " minutes " + str(seconds) + " seconds ---")
```

```
--- 11 minutes 46 seconds ---
```

[ ]:
```

```