In [417]:
```
import pandas as pd
from collections import Counter
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn import metrics

from array import *
```

Data Cleaning

In [455]:
```
data = pd.read_csv('C:/Users/Frank Shi/Desktop/ADS Project 4/compas-scores-two-ye
```

In [456]:
```
data.head()
```

Out[456]:

	id	name	first	last	compas_screening_date	sex	dob	age	age_cat	r
0	1	miguel hernandez	miguel	hernandez	14/08/2013	Male	18/04/1947	69	Greater than 45	O
1	3	kevon dixon	kevon	dixon	27/01/2013	Male	22/01/1982	34	25 - 45	Afri Amer
2	4	ed philo	ed	philo	14/04/2013	Male	14/05/1991	24	Less than 25	Afri Amer
3	5	marcu brown	marcu	brown	13/01/2013	Male	21/01/1993	23	Less than 25	Afri Amer
4	6	bouthy pierrelouis	bouthy	pierrelouis	26/03/2013	Male	22/01/1973	43	25 - 45	O

5 rows × 53 columns

In [457]:
```
### remove rows contains other races, update AA to be 1 and Cau to be 0
data = data[data["race"].str.contains("Other")==False]
data['race'] = data['race'].replace(['African-American', 'Caucasian'], [1, 0])
```

In [458]:
```
#### update vr_charge_degree to be dummy
data['vr_charge_degree']
data['vr_charge_degree'] = data['vr_charge_degree'].fillna('0')
data['vr_charge_degree'] = data['vr_charge_degree'].str.contains(pat = '0')
```

In [459]:
```
Counter(data['vr_charge_degree'])
```

Out[459]:
```
Counter({False: 781, True: 6056})
```

In [460]:
```
### DROP the following columns
### Drop the dates and columns contains to many missing values
df = data.drop(['type_of_assessment','id', 'name', 'first', 'last','compas_screen
df.head()
```

Out[460]:

	sex	age	age_cat	race	juv_fel_count	decile_score	juv_misd_count	juv_other_count	priors
1	Male	34	25 - 45	1	0	3	0	0	
2	Male	24	Less than 25	1	0	4	0	1	
3	Male	23	Less than 25	1	0	8	1	0	
6	Male	41	25 - 45	0	0	6	0	0	
8	Female	39	25 - 45	0	0	1	0	0	

5 rows × 24 columns

In [461]:
```
##fill na with 0
df['days_b_screening_arrest'].fillna(0, inplace=True)
df['c_days_from_compas'].fillna(0, inplace=True)
```

In [462]:
```
###Dummie transformation
to_dummy = ['sex', 'age_cat' ,'c_charge_degree', 'vr_charge_degree', 'v_score_tex
dummies = pd.get_dummies(df[to_dummy])
df = pd.concat([df, dummies], axis=1)
df = df.drop(to_dummy, axis=1)
```

In [463]:
```
df.isna().sum()
```

Out[463]:
```
age                        0
race                       0
juv_fel_count              0
decile_score               0
juv_misd_count             0
juv_other_count            0
priors_count               0
days_b_screening_arrest    0
c_days_from_compas         0
is_recid                   0
is_violent_recid           0
decile_score.1             0
v_decile_score             0
priors_count.1             0
start                      0
end                        0
event                      0
two_year_recid             0
sex_Female                 0
sex_Male                   0
age_cat_25 - 45            0
age_cat_Greater than 45    0
age_cat_Less than 25       0
c_charge_degree_F          0
c_charge_degree_M          0
v_score_text_High          0
v_score_text_Low           0
v_score_text_Medium        0
score_text_High            0
score_text_Low             0
score_text_Medium          0
dtype: int64
```

In [464]:
```
### divide the dataset into 2 dataset by races

df_cau = df[df["race"] == 0]
df_aa = df[df["race"] == 1]
print(df_cau.shape[0])
print(df_aa.shape[0])
print('Number of Cau race Commit a Crime in 2 years',   df_cau[df_cau["two_year_r
print( 'Number of AA race Commit a Crime in 2 years', df_aa[df_aa["two_year_recid
print('Percentage of Cau race Commit a Crime in 2 years' ,df_cau[df_cau["two_year
print('Percentage of AA race Commit a Crime in 2 years', df_aa[df_aa["two_year_re


```

```
2454
3696
Number of Cau race Commit a Crime in 2 years 966
Number of AA race Commit a Crime in 2 years 1901
Percentage of Cau race Commit a Crime in 2 years 0.39364303178484106
Percentage of AA race Commit a Crime in 2 years 0.5143398268398268
```

In [465]:
```
## drop the race column
df_cau = df_cau.drop('race', axis=1)
df_aa = df_aa.drop('race', axis=1)
```

Splitting the data for both races

In [466]:
```
#### splitting datasets
### Caucasian
X_cau = df_cau.drop("two_year_recid", axis=1)
y_cau = df_cau["two_year_recid"]
X_train_cau, X_test_cau, y_train_cau, y_test_cau = train_test_split(X_cau, y_cau,

### African American
X_aa = df_aa.drop("two_year_recid", axis=1)
y_aa = df_aa["two_year_recid"]
X_train_aa, X_test_aa, y_train_aa, y_test_aa = train_test_split(X_aa, y_aa, test_
```

base model

In [467]:
```
### merge X and y for training set
X_train_basemodel = pd.concat([X_train_cau, X_train_aa], axis=0)
y_train_basemodel = pd.concat([y_train_cau, y_train_aa], axis=0)

### Base model
basemodel = LogisticRegression()
basemodel.fit(X_train_basemodel, y_train_basemodel)

### Base model acc for AA
y_pred_aa = basemodel.predict(X_test_aa)
accuracy_aa = metrics.accuracy_score(y_test_aa, y_pred_aa)
print('Accuracy for African American:' + str(accuracy_aa))

### acc for Cau
y_pred_cau = basemodel.predict(X_test_cau)
accuracy_cau = metrics.accuracy_score(y_test_cau, y_pred_cau)
print('Accuracy for Cauasin:' + str(accuracy_cau))

### overall acc
print('Accuracy total:' , ( sum(y_pred_cau == y_test_cau) + sum(y_pred_aa == y_te

### acc difference/calibration
print('Differece/Calibration:' + str(abs(accuracy_cau-accuracy_aa)))
```

```
Accuracy for African American:0.9621621621621622
Accuracy for Cauasin:0.9877800407331976
Accuracy total: 0.9723801787164906
Differece/Calibration:0.02561787857103537

C:\Users\Frank Shi\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

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

Local Preferential Sampling method (Logitic regression)

The data splitting framework: We first split the data into 2 datasets by races. Within each race, we split the data into training data and testing data. We only applied local preferential sampling method on the 2 training datasets (One for African American and one for Caucasian). We combined the 2 updated training datasets, and build a new classifier based on this combied training dataset. Also, We combined the testing datasets from 2 races, and calcuated overall accuracy and calibrations based on this combined testing set from 2 races.

Race: African American.

In [468]:
```
###data split for 2 races

### Caucasian
X_cau = df_cau.drop("two_year_recid", axis=1)
y_cau = df_cau["two_year_recid"]
X_train_cau, X_test_cau, y_train_cau, y_test_cau = train_test_split(X_cau, y_cau,

### African American
X_aa = df_aa.drop("two_year_recid", axis=1)
y_aa = df_aa["two_year_recid"]
X_train_aa, X_test_aa, y_train_aa, y_test_aa = train_test_split(X_aa, y_aa, test_
```

In [469]:
```
### Initial logistic regression on training data for African American
log_aa = LogisticRegression()
log_aa.fit(X_train_aa, y_train_aa)

y_pred_aa = log_aa.predict(X_test_aa)
accuracy = metrics.accuracy_score(y_test_aa, y_pred_aa)
print('ACC for AA without resampling: ', accuracy) ### AA represents African Amer
```

```
ACC for AA without resampling:  0.95

C:\Users\Frank Shi\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

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

In [470]:
```
## probability table construction
## dd is the table contains logits
dd = log_aa.predict_proba(X_train_aa)
(abs(dd[1,0] - dd[1,1])) ### check the difference of the second row
```

Out[470]:
```
0.999028204974056
```

In [471]:
```
## calculate the logit differences from previous table
logit_diff = []
for i in range(len(dd)):
    logit_diff.append(abs(dd[i,0] - dd[i,1]))  ### take the absolute value
```

In [472]:
```
np.array(logit_diff)[np.array(logit_diff) <= 0.4]   ###max logit = 0.65
print(len(np.array(logit_diff)[np.array(logit_diff) <= 0.4])) ### number of logit

### A list contains Trues and Falses, length of the list equals to number of rows
position = np.array(logit_diff) <= 0.4
```

```
198
```

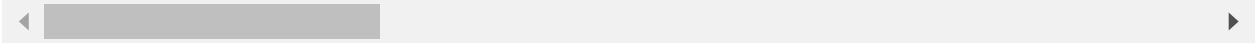
In [473]:
```
##Gathering index for True and False
selected_rows = []  #### index with True
not_selected_rows = [] ### index with False

for i in range(len(position)):
    if position[i] == True :
        selected_rows.append(i)
    else:
        not_selected_rows.append(i)
```

In [474]:
```
### rows with distance below threshold
selected_X = X_train_aa.iloc[selected_rows, ] ### with true in positions
selected_y = y_train_aa.iloc[selected_rows, ]

### rows with distance above threshold
unselected_X = X_train_aa.iloc[not_selected_rows, ] ### with true in positions
unselected_y = y_train_aa.iloc[not_selected_rows, ]

### merge X and y for selected and unselected
selected = pd.concat([selected_X, selected_y], axis=1)
unselected = pd.concat([unselected_X, unselected_y], axis=1)

### Only keep rows from selected that has two_year_recid == 0
### Duplicate kept rows by c
selected = selected[selected.two_year_recid == 0] #######remain the labels with 0
repeated = pd.concat([selected]*4, ignore_index=True) #### duplicate the rows  4


### merge duplicated rows and unselected rows vertically
df_aa_train_new = pd.concat([unselected, repeated], axis=0)
(df_aa_train_new)
```

Out[474]:

	age	juv_fel_count	decile_score	juv_misd_count	juv_other_count	priors_count	days_b_scre
5221	33	0	1	0	0	0	
235	52	0	4	0	0	0	
5569	50	0	9	0	0	9	
5601	25	0	5	0	0	0	
1371	28	0	4	0	1	2	
...	...	...	...	...	...	...	
495	26	0	9	0	0	0	
496	23	0	4	0	0	0	
497	50	1	2	0	0	5	
498	25	0	7	0	0	5	
499	30	0	2	0	1	3	

3258 rows × 30 columns



In [475]:
```
print(df_aa_train_new.shape[0])
print( 'Number of AA race Commit a Crime in 2 years after applied local sampling'
print('Percentage of AA race Commit a Crime in 2 years after applied local sampli
```

```
3258
Number of AA race Commit a Crime in 2 years after applied local sampling 1476
Percentage of AA race Commit a Crime in 2 years after applied local sampling 0.
4530386740331492
```

In [476]:
```
print(Counter(y_train_aa))
Counter(y_train_cau)
770/(1184 +770)
```

```
Counter({1: 1549, 0: 1407})
```

Out[476]:
```
0.3940634595701126
```

In [477]:
```
print(Counter(y_train_aa))
1519/(1519+1437)
```

```
Counter({1: 1549, 0: 1407})
```

Out[477]:
```
0.5138700947225981
```

Local resampling on Causian

In [478]:
```
### M Cau
log_cau = LogisticRegression()
log_cau.fit(X_train_cau, y_train_cau)

y_pred_cau = log_cau.predict(X_test_cau)
accuracy = metrics.accuracy_score(y_test_cau, y_pred_cau)
print('Acc for Cau wihtout resampling',accuracy)

dd = log_cau.predict_proba(X_train_cau)

## calculate the logit differences
logit_diff = []
for i in range(len(dd)):
    logit_diff.append(abs(dd[i,0] - dd[i,1])) ### take the absolute value

np.array(logit_diff)[np.array(logit_diff) <= 0.3]   ###max logit = 0.65
print(len(np.array(logit_diff)[np.array(logit_diff) <= 0.3])) ### number of logit
position = np.array(logit_diff) <= 0.3


##Gathering index for True and False
selected_rows = []  ##Gathering index for True
not_selected_rows = [] ##Gathering index for False
for i in range(len(position)):
    if position[i] == True :
        selected_rows.append(i)
    else:
        not_selected_rows.append(i)

selected_X = X_train_cau.iloc[selected_rows, ] ### with true in positions
selected_y = y_train_cau.iloc[selected_rows, ]

unselected_X = X_train_cau.iloc[not_selected_rows, ] ### with true in positions
unselected_y = y_train_cau.iloc[not_selected_rows, ]

selected = pd.concat([selected_X, selected_y], axis=1)
unselected = pd.concat([unselected_X, unselected_y], axis=1)

### Only keep rows from selected that has two_year_recid == 1
### Duplicate kept rows by c
selected = selected[selected.two_year_recid == 1] #######remain the labels with 1
repeated = pd.concat([selected]*5, ignore_index=True)


### merge duplicated rows and unselected rows vertically
df_cau_train_new = pd.concat([unselected, repeated], axis=0)
(df_cau_train_new)
```

```
Acc for Cau wihtout resampling 0.9653767820773931
37

C:\Users\Frank Shi\anaconda3\lib\site-packages\sklearn\linear_model\_logisti
c.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
```

```
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

Out[478]:

	age	juv_fel_count	decile_score	juv_misd_count	juv_other_count	priors_count	days_b_scre
6186	24	0	5	0	0	3	
6259	52	0	1	0	0	6	
6621	20	0	3	0	0	0	
855	26	0	4	0	0	0	
275	25	0	5	0	0	0	
...	...	...	...	...	...	...	
75	47	0	5	0	0	4	
76	59	0	2	0	0	0	
77	66	0	1	0	0	1	
78	64	0	10	0	0	8	
79	43	0	1	0	0	0	

2006 rows × 30 columns

In [479]:
```
print(df_cau_train_new.shape[0])
print( 'Number of CAU race Commit a Crime in 2 years after applied local sampling
print('Percentage of CAU race Commit a Crime in 2 years after applied local sampl
```

```
2006
Number of CAU race Commit a Crime in 2 years after applied local sampling 837
Percentage of CAU race Commit a Crime in 2 years after applied local sampling
0.4172482552342971
```

Local preferential massaing: Calculate the Overall ACC and Calibration

In [480]:
```
### merge the new training sets
df_train_new_total = pd.concat([df_aa_train_new, df_cau_train_new], axis=0)
df_train_new_x =  df_train_new_total.drop("two_year_recid", axis=1)
df_train_new_y = df_train_new_total["two_year_recid"]


model2 = LogisticRegression()
model2.fit(df_train_new_x, df_train_new_y)
```

```
C:\Users\Frank Shi\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

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

Out[480]:
```
LogisticRegression()
```

In [481]:
```
### acc for AA
y_pred_aa = model2.predict(X_test_aa)
accuracy_aa = metrics.accuracy_score(y_test_aa, y_pred_aa)
print('Accuracy for African American:' + str(accuracy_aa))

### acc for Cau
y_pred_cau = model2.predict(X_test_cau)
accuracy_cau = metrics.accuracy_score(y_test_cau, y_pred_cau)
print('Accuracy for Cauasin:' + str(accuracy_cau))

### overall acc
print('Accuracy total:' , ( sum(y_pred_cau == y_test_cau) + sum(y_pred_aa == y_te

### difference (calibrition score)
print('Difference (calibrition score):' + str(abs(accuracy_cau -accuracy_aa)))
```

```
Accuracy for African American:0.9608108108108108
Accuracy for Cauasin:0.9572301425661914
Accuracy total: 0.9593826157595451
Difference (calibrition score):0.003580668244619334
```

Local Massaging (logistic)

local massage for African American (AA)

In [482]:
```
Counter(y_train_aa)
```

Out[482]:
```
Counter({0: 1407, 1: 1549})
```

In [483]:
```
print('% of AA race Commit a Crime in 2 years before apply local massage', Counte
```

```
% of AA race Commit a Crime in 2 years before apply local massage 0.52401894451
9621
```

In [484]:
```
### Method 1: local massage for African American

## table contains 2 logits per row
table_aa = log_aa.predict_proba(X_train_aa)

###calculate the abs difference between 2 logits from above table
logit_diff_aa = []
for i in range(len(table_aa)):
    logit_diff_aa.append(abs(table_aa[i,0] - table_aa[i,1]))

print('Number of obervations below threshold',len(np.array(logit_diff_aa)[np.arra
#### a list contains trues and falses
position_aa = np.array(logit_diff_aa) <= 0.6


##label update:if the index corresopding to true, we update the lable to 0
for i in range(len(position_aa)):
    if position_aa[i] == True :
        y_train_aa.iloc[i] =0

print(X_train_aa.shape)
print(len(y_train_aa))
```

```
Number of obervations below threshold 301
(2956, 29)
2956
```

In [485]:
```
Counter(y_train_aa) ##2956
```

Out[485]:
```
Counter({0: 1527, 1: 1429})
```

In [486]:
```
print('Number of AA race Commit a Crime in 2 years after applied local massage',
```

```
Number of AA race Commit a Crime in 2 years after applied local massage 0.48342
35453315291
```

local massage for Cau

In [487]:
```
### Method 2 on Cau
y_pred_cau = log_cau.predict(X_test_cau)
table_cau = log_cau.predict_proba(X_train_cau)

## calculate the logit differences
logit_diff_cau = []
for i in range(len(table_cau)):
    logit_diff_cau.append(abs(table_cau[i,0] - table_cau[i,1]))

print('Number of obervations below threshold', len(np.array(logit_diff_cau)[np.ar
position_cau = np.array(logit_diff_cau) <= 0.5


##label update
for i in range(len(position_cau)):
    if position_cau[i] == True :
        y_train_cau.iloc[i] =1

print(X_train_cau.shape)
print(len(y_train_cau))
```

```
Number of obervations below threshold 65
(1963, 29)
1963
```

In [488]:
```
Counter(y_train_cau) ##1960
```

Out[488]:
```
Counter({0: 1156, 1: 807})
```

In [489]:
```
print('Number of Cau race Commit a Crime in 2 years after applied local massage'
```

```
Number of Cau race Commit a Crime in 2 years after applied local massage 0.4111
0545084055017
```

In [490]:
```
pd.concat([y_train_aa, y_train_cau], axis=0)
```

Out[490]:
```
5221    0
235     0
5569    0
5601    0
1371    1
       ..
3416    0
4088    0
6544    0
2884    1
3776    1
Name: two_year_recid, Length: 4919, dtype: int64
```

Overall acc and calibration

In [491]:
```
### merge the new training sets
X_total_new = pd.concat([X_train_aa, X_train_cau], axis=0)
y_total_new = pd.concat([y_train_aa, y_train_cau], axis=0)

model3 = LogisticRegression()
model3.fit(X_total_new, y_total_new)
```

```
C:\Users\Frank Shi\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

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

Out[491]:
```
LogisticRegression()
```

In [492]:
```
### acc for AA
y_pred_aa = model3.predict(X_test_aa)
accuracy_aa = metrics.accuracy_score(y_test_aa, y_pred_aa)
print('Accuracy for African American:' + str(accuracy_aa))

### acc for Cau
y_pred_cau = model3.predict(X_test_cau)
accuracy_cau = metrics.accuracy_score(y_test_cau, y_pred_cau)
print('Accuracy for Cauasin:' + str(accuracy_cau))

### overall acc
print('Accuracy total:' , ( sum(y_pred_cau == y_test_cau) + sum(y_pred_aa == y_te

### acc difference/calibration
print('Differece/Calibration:' + str(abs(accuracy_cau-accuracy_aa)))
```

```
Accuracy for African American:0.95
Accuracy for Cauasin:0.9490835030549898
Accuracy total: 0.9496344435418359
Differece/Calibration:0.0009164969450101701
```