BUSINESS ANALYTICS 6TH ASSIGNMENT

16102275 Park Hyun Woo

CONTEXT

1. DATA PARTITION
2. VARIABLE SELECTION
3. MODELING

DATA PARTITIONING

DATA PARTITION

```
Pref_Dem2012 = data.loc[data['per_dem_2012']>data['per_gop_2012']]
Pref_Dem2016 = data.loc[data['per_dem_2016']>data['per_gop_2016']]

Pref_Rep2012 = data.loc[data['per_dem_2012']<data['per_gop_2012']]
Pref_Rep2016 = data.loc[data['per_dem_2016']<data['per_gop_2016']]

Pref_Rep2012['per_point_diff_2012'] = abs(Pref_Rep2012['per_point_diff_2012'])
Pref_Rep2016['per_point_diff_2016'] = abs(Pref_Rep2016['per_point_diff_2016'])
```

- Firstly I partition data , standard is in each year what PARTY is selected in that County
- In each year I make prefer demo and prefer republic data partition.
- And per_point_diff columns is per_gop - per_dem ,so some value have negative. So for better understanding when I select variable , I change that to absolute value

DATA PARTITION

```
High_Pref_Dem2012 = Pref_Dem2012.loc[Pref_Dem2012['per_point_diff_2012']>
                                     Pref_Dem2012['per_point_diff_2012'].mean()]
High_Pref_Rep2012 = Pref_Rep2012.loc[Pref_Rep2012['per_point_diff_2012']>
                                    Pref_Rep2012['per_point_diff_2012'].mean()]
High_Pref_Dem2016 = Pref_Dem2016.loc[Pref_Dem2016['per_point_diff_2016']>
                                     Pref_Dem2016['per_point_diff_2016'].mean()]
High_Pref_Rep2016 = Pref_Rep2016.loc[Pref_Rep2016['per_point_diff_2016']>
                                    Pref_Rep2016['per_point_diff_2016'].mean()]
```

- I want to find variables that make a big difference between the Republican region and the Democratic region.
- So I partition previous prefer democracy and prefer republic to high prefer democracy and high prefer republic.

VARIABLE SELECTION

ABOUT SEX&RACE&POPULATION

```
# 1st checking relationship between race & sex ~ prefercence Party
# RHI125214 - White alone, percent, 2014
# RHI225214 - Black or African American alone, percent, 2014
# RHI325214 - American Indian and Alaska Native alone, percent, 2014
# RHI425214 - Asian alone, percent, 2014
# RHI525214 - Native Hawaiian and Other Pacific Islander alone, percent, 2014
# RHI625214 - Two or More Races, percent, 2014
# RHI725214 - Hispanic or Latino, percent, 2014
# RHI825214 - White alone, not Hispanic or Latino, percent, 2014

# PST045214 - Population, 2014 estimate
# PST040210 - Population, 2010 (April 1) estimates base
# PST120214 - Population, percent change - April 1, 2010 to July 1, 2014
# POP010210 - Population, 2010
# POP645213 - Foreign born persons, percent, 2009-2013
# POP715213 - Living in same house 1 year & over, percent, 2009-2013
```

- **Among above variables , I think 2014 years data can be more meaningful 2016 election data set. 2010year and 2009-2013 years data can be more meaningful 2012 election data set.**
- **To select variable, race and sex variables are used in 2016 , population data is used in in 2012**
- **meaning of the RHI125214 is include both RHI725214 and RHI825214 so i don't use RHI125214**

ABOUT SEX&RACE&POPULATION

```
race = ['RHI125214', 'RHI225214', 'RHI325214', 'RHI425214','RHI525214', 'RHI625214', 'RHI7252

corr_Dem2012 = High_Pref_Dem2012[race+['SEX255214','PST040210','POP645213','POP715213','per_p
corr_Rep2012 = High_Pref_Rep2012[race+['SEX255214','PST040210','POP645213','POP715213','per_p
corr_Dem2016 = High_Pref_Dem2016[race+['SEX255214','PST045214','POP645213','POP715213','per_p
corr_Rep2016 = High_Pref_Rep2016[race+['SEX255214','PST045214','POP645213','POP715213','per_p


diff_sex_race_average2012 = High_Pref_Dem2012[race+['SEX255214','PST040210','POP645213','POP7
- High_Pref_Rep2012[race+['SEX255214','PST040210']].mean()

diff_sex_race_average2016 = High_Pref_Dem2016[race+['SEX255214','PST045214','POP645213','POP7
- High_Pref_Rep2016[race+['SEX255214','PST045214']].mean()

corr_Dem2012.rename(columns = {'per_point_diff_2012':'dem_corr_per_point_diff_2012'},inplace=
corr_Rep2012.rename(columns = {'per_point_diff_2012':'rep_corr_per_point_diff_2012'},inplace=
corr_Dem2016.rename(columns = {'per_point_diff_2016':'dem_corr_per_point_diff_2016'},inplace=
corr_Rep2016.rename(columns = {'per_point_diff_2016':'rep_corr_per_point_diff_2016'},inplace=

result = pd.DataFrame([corr_Dem2012['dem_corr_per_point_diff_2012'],
                       corr_Rep2012['rep_corr_per_point_diff_2012'],
                       corr_Dem2016['dem_corr_per_point_diff_2016'],
                       corr_Rep2016['rep_corr_per_point_diff_2016'],
                       diff_sex_race_average2012,
                       diff_sex_race_average2016])
result = result.fillna(0)
result['PST040210'] = result['PST040210']+result['PST045214']
result.drop(columns = ['PST045214'],inplace = True)
result = result.transpose()

result.rename(index = {'PST040210':'Population'},columns = {'Unnamed 0': 'diff_average2012','

print('Race&Sex&Population result in 2016 vote election')
result
```

- Correlation between per_point different and difference of each variable's mean is used in my variable selection.
- All of difference of variables is democracy minus republican
- And I make new population column because PST045214 PST040210 represent population of each year

ABOUT SEX&RACE&POPULATION

	dem_corr_per_point_diff_2012	rep_corr_per_point_diff_2012	dem_corr_per_point_diff_2016	rep_corr_per_point_diff_2016	diff_average2012	diff_average2016
RHI225214	0.182355	-0.225903	0.136992	-0.209210	26.475093	27.438119
RHI325214	0.155258	-0.081659	0.025468	-0.042861	4.489591	4.474257
RHI425214	0.135091	-0.124238	0.126916	-0.182547	3.761338	4.923762
RHI525214	0.090462	0.024889	-0.051708	-0.048821	0.266543	0.348515
RHI625214	0.046876	-0.134582	-0.054628	-0.088305	2.413011	2.621782
RHI725214	0.206932	0.145934	0.084923	0.083763	16.071375	19.103465
RHI825214	-0.485999	0.013026	-0.269674	0.051523	48.473606	43.254950
SEX255214	0.175440	-0.100083	0.138210	-0.064374	50.652045	51.002970
Population	0.092761	-0.173246	0.124200	-0.279434	319767.583643	431029.148515
POP645213	0.171305	0.138767	0.211444	0.062088	9.482156	11.851980
POP715213	0.090471	0.068954	0.035612	0.128171	85.374349	84.589604

- dem_correlation high and rep_correlation low (difference sign is best) and differ average is large - > that variable can show difference very well
- There block Rectangle mean more meaning year full in that variables
- Among those variables , different sign of correlation in same year and high different average(those value almost percent value) mean meaning variables for showing difference
- RHI825214 and SEX255214 is meaningful value so I select those values

ABOUT AGE VARIABLE

```
# AGE135214 - Persons under 5 years, percent, 2014
# AGE295214 - Persons under 18 years, percent, 2014
# AGE775214 - Persons 65 years and over, percent, 2014
# because age data in year2016 -> So used only 2016 vote result
# for calculating correlation and difference of average
# i make new column percent of age between 18 to 64
```

- **Those are all 2014 data so I think more meaningful in2016 vote selection**

ABOUT AGE VARIABLE

	dem_corr_per_point_diff_2016	rep_corr_per_point_diff_2016	diff_average2016
AGE135214	0.131556	0.040791	0.592094
AGE295214	-0.022188	0.052504	0.103379
AGE775214	-0.184082	0.143362	-4.476632
AGE18_64	0.099188	-0.225647	3.781159

- **According to above table , Age7725214 is best, pigsties differ_average and**

ABOUT AGE VARIABLE

```
age = ['AGE135214','AGE295214','AGE775214']

High_Pref_Dem2016['AGE18_64'] = 100 - (High_Pref_Dem2016[age[0]]+High_Pref_Dem2016[age[1]]+High_Pref_Dem2016[age[2]])
High_Pref_Rep2016['AGE18_64'] = 100 - (High_Pref_Rep2016[age[0]]+High_Pref_Rep2016[age[1]]+High_Pref_Rep2016[age[2]])
age = age+['AGE18_64']

corr_Dem2016 = High_Pref_Dem2016[age+['SEX255214','per_point_diff_2016']].corr()[:4]
corr_Rep2016 = High_Pref_Rep2016[age+['SEX255214','per_point_diff_2016']].corr()[:4]

corr_Dem2016.rename(columns = {'per_point_diff_2016':'dem_corr_per_point_diff_2016'},inplace=True)
corr_Rep2016.rename(columns = {'per_point_diff_2016':'rep_corr_per_point_diff_2016'},inplace=True)



diff_age_average2016 = High_Pref_Dem2016[age].mean() - High_Pref_Rep2016[age].mean()

result = pd.DataFrame([corr_Dem2016['dem_corr_per_point_diff_2016'],corr_Rep2016['rep_corr_per_point_diff_2016'],diff
result = result.transpose()
result.rename(columns = {'Unnamed 0': 'diff_average2016'},inplace=True)

print('Age result in 2016 vote election')
result
```

- **Between 18 to 24 data is not present so I make new column**
- **Above code process is same with previous variable selection**

ABOUT EDUCATION VARIABLE AND 2012 APPLIED VARIABLE

```
# POP060210 - 인구밀집도
# LND110210 - Land area in square miles, 2010
# POP815213 - Language other than English spoken at home, pct age 5+, 2009-2013
# EDU635213 - High school graduate or higher, percent of persons age 25+, 2009-2013
# EDU685213 -Bachelor's degree or higher, percent of persons age 25+, 2009-2013
# Above 3 data can usesd in 2012 only because interval is 2009 to 2013
```

```
edu_others = ['POP060210','LND110210','POP815213','EDU635213','EDU685213']
corr_Dem2012 = High_Pref_Dem2012[edu_others+['per_point_diff_2012']].corr()[:len(edu_others)]
corr_Rep2012 = High_Pref_Rep2012[edu_others+['per_point_diff_2012']].corr()[:len(edu_others)]

corr_Dem2012.rename(columns = {'per_point_diff_2012':'dem_corr_per_point_diff_2012'},inplace=True)
corr_Rep2012.rename(columns = {'per_point_diff_2012':'rep_corr_per_point_diff_2012'},inplace=True)

diff_edu_others_average2012 = High_Pref_Dem2012[edu_others].mean() - High_Pref_Rep2012[edu_others].mean()

result = pd.DataFrame([corr_Dem2012['dem_corr_per_point_diff_2012'],corr_Rep2012['rep_corr_per_point_diff_2012'],diff
result = result.transpose()
result.rename(columns = {'Unnamed 0': 'diff_average2012'},inplace=True)

print('Education&Others result in 2012 vote election')
result
```

- **Between 2009 to 2013 and 2010 data , is selected in 2012 vote election**

ABOUT EDUCATION VARIABLE AND 2012 APPLIED VARIABLE

Education&Others result in 2012 vote election

	dem_corr_per_point_diff_2012	rep_corr_per_point_diff_2012	diff_average2012
POP060210	0.316682	-0.173931	1509.470802
LND110210	-0.023904	0.103395	-113.840957
POP815213	0.246394	0.112605	10.652564
EDU635213	-0.222494	0.065575	-1.044108
EDU685213	0.001393	0.082959	8.498965

- **In this table I use only POP05210 is used because correlation difference is clear**

ABOUT INCOME AND EMPLOYMENT VARIABLES

```
# VET605213 - Veterans, 2009-2013
# LFE305213 - Mean travel time to work (minutes), workers age 16+, 2009-2013
# INC910213 - Per capita money income in past 12 months (2013 dollars), 2009-2013
# INC110213 - Median household income, 2009-2013
# PVY020213 - Persons below poverty level, percent, 2009-2013
# BZA010213 - Private nonfarm establishments, 2013
# BZA110213 - Private nonfarm employment, 2013
# BZA115213 - Private nonfarm employment, percent change, 2012-2013
# NES010213 - Nonemployer establishments, 2013
```

- **2013 data set is ambiguous whether to apply it in 2012 or 2016, so it was applied to each year**

ABOUT INCOME AND EMPLOYMENT VARIABLES

```
inc_empl= ['VET605213','LFE305213','BZA010213','BZA110213','NES010213','INC910213','INC110213','PVY020213']
#VET,BZA person number, so when comparing average -> hard to how much difference -> So i change to percent
High_Pref_Dem2012['VET605213'] = (High_Pref_Dem2012['VET605213']/High_Pref_Dem2012['POP010210'])*100
High_Pref_Rep2012['VET605213'] = (High_Pref_Rep2012['VET605213']/High_Pref_Rep2012['POP010210'])*100
High_Pref_Dem2016['VET605213'] = (High_Pref_Dem2016['VET605213']/High_Pref_Dem2016['POP010210'])*100
High_Pref_Rep2016['VET605213'] = (High_Pref_Rep2016['VET605213']/High_Pref_Rep2016['POP010210'])*100

High_Pref_Dem2012['BZA110213'] = (High_Pref_Dem2012['BZA110213']/High_Pref_Dem2012['POP010210'])*100
High_Pref_Rep2012['BZA110213'] = (High_Pref_Rep2012['BZA110213']/High_Pref_Rep2012['POP010210'])*100
High_Pref_Dem2016['BZA110213'] = (High_Pref_Dem2016['BZA110213']/High_Pref_Dem2016['POP010210'])*100
High_Pref_Rep2016['BZA110213'] = (High_Pref_Rep2016['BZA110213']/High_Pref_Rep2016['POP010210'])*100


corr_Dem2012 = High_Pref_Dem2012[inc_empl+['per_point_diff_2012']].corr()[:len(inc_empl)]
corr_Rep2012 = High_Pref_Rep2012[inc_empl+['per_point_diff_2012']].corr()[:len(inc_empl)]
corr_Dem2016 = High_Pref_Dem2016[inc_empl+['per_point_diff_2016']].corr()[:len(inc_empl)]
corr_Rep2016 = High_Pref_Rep2016[inc_empl+['per_point_diff_2016']].corr()[:len(inc_empl)]


corr_Dem2012.rename(columns = {'per_point_diff_2012':'dem_corr_per_point_diff_2012'},inplace=True)
corr_Rep2012.rename(columns = {'per_point_diff_2012':'rep_corr_per_point_diff_2012'},inplace=True)
corr_Dem2016.rename(columns = {'per_point_diff_2016':'dem_corr_per_point_diff_2016'},inplace=True)
corr_Rep2016.rename(columns = {'per_point_diff_2016':'rep_corr_per_point_diff_2016'},inplace=True)

diff_inc_empl_average2012 = High_Pref_Dem2012[inc_empl].mean() - High_Pref_Rep2012[inc_empl].mean()
diff_inc_empl_average2016 = High_Pref_Dem2016[inc_empl].mean() - High_Pref_Rep2016[inc_empl].mean()

result = pd.DataFrame([corr_Dem2012['dem_corr_per_point_diff_2012'],
                       corr_Rep2012['rep_corr_per_point_diff_2012'],
                       corr_Dem2016['dem_corr_per_point_diff_2016'],
                       corr_Rep2016['rep_corr_per_point_diff_2016'],
                       diff_inc_empl_average2012,
                       diff_inc_empl_average2016])
result = result.transpose()
result.rename(columns = {'Unnamed 0': 'diff_average2012','Unnamed 1': 'diff_average2016'},inplace=True)

print('Income&Employment result in 2012 vote election')
result
```

```
data['INC910213'] = (data['INC910213']/data['INC910213'].max())* 100
data['INC110213'] = (data['INC110213']/data['INC110213'].max())* 100
```

- **VET605213 and BZA110213 is number variable so I change that variable to percent variable**
- **And income also number variable and I want to see that number income level so I change ,in my code's first part ,those variables to income level percetage**

ABOUT INCOME AND EMPLOYMENT VARIABLES

	dem_corr_per_point_diff_2012	rep_corr_per_point_diff_2012	dem_corr_per_point_diff_2016	rep_corr_per_point_diff_2016	diff_average2012	diff_average2016
VET605213	-0.074265	0.057908	-0.080730	0.149987	-0.004611	-0.008641
LFE305213	0.147855	-0.185062	0.246388	-0.143512	1.064067	0.976849
BZA010213	0.089225	-0.152350	0.137682	-0.270568	7581.250141	10341.104683
BZA110213	-0.068912	0.157553	-0.088538	0.238477	-0.000670	-0.007981
NES010213	0.091184	-0.126085	0.121689	-0.235772	24831.398041	34518.295129
INC910213	-0.068625	0.121061	0.123475	0.087211	2.878124	5.526776
INC110213	-0.130169	0.118910	0.032497	-0.000742	1.253961	4.034725
PVY020213	0.311761	-0.145193	0.132606	-0.052317	5.257481	5.264090

- **PVY020213 and NES010123 is selected in this table. Different sign value and high difference of average.**

MODELING

MODELING

```
variable_list
```

```
['AGE775214', 'RHI825214', 'SEX255214', 'POP060210', 'PVY020213', 'NES010213']
```

```
variable_set = list(itertools.combinations(variable_list,3))

for i in range(len(variable_set)):
    variable_set[i] = list(variable_set[i])
```

- Those are selected variables
- And I want to see combination of 3 , and by each combination set I want to see score value. So I make variable set

MODELING

```
data.reset_index(drop=True,inplace=True)

logi = LogisticRegression(C=1)
kfold = KFold(n_splits=5,shuffle=True)

score_list = []
val_score_list = []
target = data['target']

report_set = []
#these 3 list are used after filter variable sets for representing classfication report and roc curve
y_true = []
y_pred = []
y_proba = []
for i in variable_set:
    train_X,test_X,train_y,test_y = train_test_split(data[i], target, test_size=0.2, stratify=target, random_state=
    logi.fit(train_X,train_y)
    score_list.append(logi.score(test_X,test_y))
    each_val_score = []
    y_true.append(test_y)
    y_pred.append(logi.predict(test_X))
    y_proba.append(logi.predict_proba(test_X))

    for train_idx, test_idx in kfold.split(data):
        train_X, val_X = data[i].iloc[train_idx],data[i].iloc[test_idx]
        train_y, val_y = target[train_idx],target[test_idx]
        logi.fit(train_X,train_y)
        each_val_score.append(logi.score(val_X,val_y))



    val_score_list.append(each_val_score)
```

- For filtering those many variable combination set, I use 5 K-Fold validation score value so I make that code.
- y_true , y_pred, y_proba lists are made for after filtering variable set ,to represent classification and ROC curve

MODELING

```
mean_val_score = []
for i in range(len(val_score_list)):
    mean_val_score.append(np.mean(val_score_list[i]))
print("max %.4f" %(max(mean_val_score)))
mean_val_score
```

```
max 0.9068

[0.8882066342198573,
 0.9017011349501685,
 0.8776080886489499,
 0.8869302668861903,
 0.8740783368515584,
 0.8599438460307711,
 0.8721485602803568,
 0.8837117360763447,
 0.8785670415425827,
 0.8808152647959,
 0.9068427327576863,
 0.8878840585694159,
 0.8984903459559336,
 0.9036386533369806,
 0.8926984356371255,
 0.8956021326121402,
 0.8814717707596786,
 0.8763368825256901,
 0.8699003370270395,
 0.8737475032644655]
```

```
top_index = []
mean_val_score
for i in range(len(variable_set)):
    A = mean_val_score[i]
    if A>0.89:
        top_index.append(mean_val_score.index(A))
top_index
```

```
[1, 10, 12, 13, 14, 15]
```

- This is result of average of 5 K-FOLD score .
- I filter variable set over 0.89.
- So those top index of variable set is have over 0.89 average of score
- Below shows the variable sets

```
new_variable_set = []
for i in top_index:
    new_variable_set.append(variable_set[i])
new_variable_set
```

```
[['AGE775214', 'RHI825214', 'POP060210'],
 ['RHI825214', 'SEX255214', 'POP060210'],
 ['RHI825214', 'SEX255214', 'NES010213'],
 ['RHI825214', 'POP060210', 'PVY020213'],
 ['RHI825214', 'POP060210', 'NES010213'],
 ['RHI825214', 'PVY020213', 'NES010213']]
```

MODELING

```
var_index = 0
for i in top_index:
    print(new_variable_set[var_index],"mean_score : " ,mean_val_score[i],"\n",
          classification_report(y_true[i],y_pred[i]))
    var_index = var_index + 1
```

1

```
['AGE775214', 'RHI825214', 'POP060210'] score :  0.9017011349501685
              precision    recall  f1-score   support

           0       0.91      0.98      0.94       525
           1       0.81      0.45      0.58        98

    accuracy                           0.90       623
   macro avg       0.86      0.71      0.76       623
weighted avg       0.89      0.90      0.88       623
```

2

```
['RHI825214', 'SEX255214', 'POP060210'] score :  0.9068427327576863
              precision    recall  f1-score   support

           0       0.91      0.98      0.94       525
           1       0.82      0.50      0.62        98

    accuracy                           0.90       623
   macro avg       0.86      0.74      0.78       623
weighted avg       0.90      0.90      0.89       623
```

3

```
['RHI825214', 'SEX255214', 'NES010213'] score :  0.8984903459559336
              precision    recall  f1-score   support

           0       0.90      0.98      0.94       525
           1       0.80      0.44      0.57        98

    accuracy                           0.89       623
   macro avg       0.85      0.71      0.75       623
weighted avg       0.89      0.89      0.88       623
```

4

```
['RHI825214', 'POP060210', 'PVY020213'] score :  0.9036386533369806
              precision    recall  f1-score   support

           0       0.91      0.98      0.94       525
           1       0.83      0.45      0.58        98

    accuracy                           0.90       623
   macro avg       0.87      0.72      0.76       623
weighted avg       0.89      0.90      0.89       623
```

5

```
['RHI825214', 'POP060210', 'NES010213'] score :  0.8926984356371255
              precision    recall  f1-score   support

           0       0.89      0.99      0.94       525
           1       0.87      0.35      0.50        98

    accuracy                           0.89       623
   macro avg       0.88      0.67      0.72       623
weighted avg       0.89      0.89      0.87       623
```

6

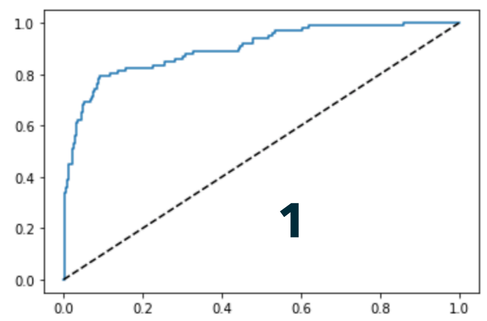
```
['RHI825214', 'PVY020213', 'NES010213'] score :  0.8956021326121402
              precision    recall  f1-score   support

           0       0.89      0.99      0.94       525
           1       0.86      0.37      0.51        98

    accuracy                           0.89       623
   macro avg       0.88      0.68      0.73       623
weighted avg       0.89      0.89      0.87       623
```

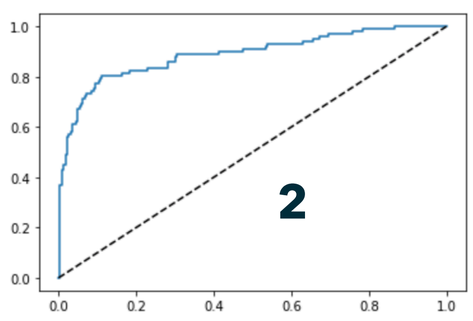
MODELING

['AGE775214', 'RHI825214', 'POP060210']



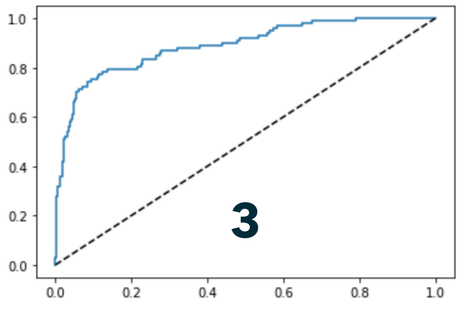
roc score is 0.8993

['RHI825214', 'SEX255214', 'POP060210']



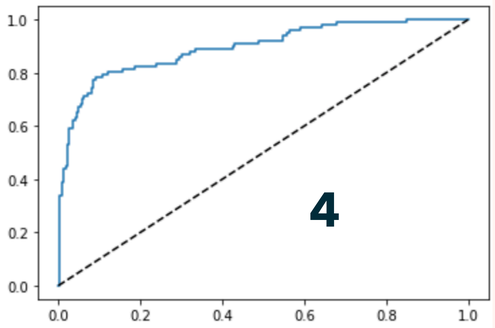
roc score is 0.8881

['RHI825214', 'SEX255214', 'NES010213']



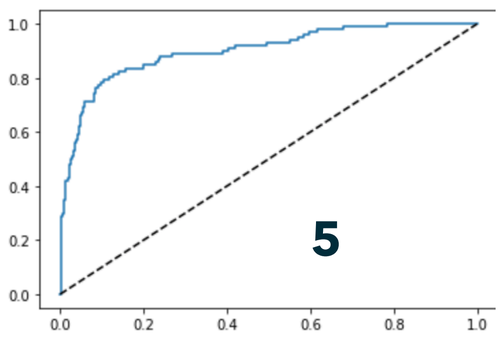
roc score is 0.8889

['RHI825214', 'POP060210', 'PVY020213']



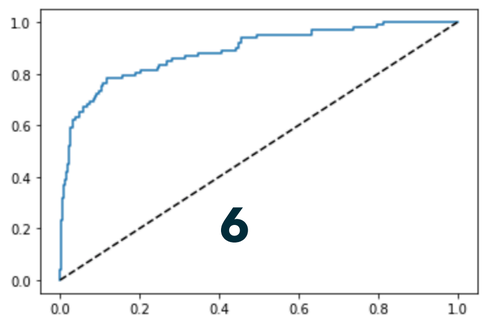
roc score is 0.8945

['RHI825214', 'POP060210', 'NES010213']



roc score is 0.8996

['RHI825214', 'PVY020213', 'NES010213']



roc score is 0.8876

- By using macro average, weighted average-> 1,2,3,4
- By using roc score -> 1,4,5
- I select 1st variable set
- ['AGE775214','RHI825214','PO 6

```
#for roc curve AUROC value
xx = np.linspace(0,1,10)
var_index = 0
for i in top_index:
    fpr, tpr, thresholds = roc_curve(y_true[i],y_proba[i][:,1],pos_label=1)
    print("\n",new_variable_set[var_index])
    plt.plot(fpr,tpr)
    plt.plot(xx,xx,'k--')#y = x graph
    plt.show()
    print("\nroc score is %.4f" %roc_auc_score(y_true[i],y_proba[i][:,1]))
    var_index = var_index+1
```

RESULT

- ['AGE775214','RHI825214','POP60201'] is best variable set in my selection
- AGE775214 - Persons 65 years and over, percent, 2014
- RHI825214 - White alone, not Hispanic or Latino, percent, 2014
- POP060210 - population density
- SO I can say Person who age 65 and over and person who white person not Hispaninc or Latino and population density is important to presidential election
- And person who age 65 and over and person who white person more prefer Republic Party than Democracy Party according to my variable selection procedure and person who white person
- And high population density location more prefer Democracy Party than Republic Party

RHI825214	-0.485999	0.013026	-0.269674	0.051523	48.473606	43.254950
POP060210		0.316682		-0.173931	1509.470802	
AGE775214		-0.184082		0.143362	-4.476632	

THANK YOU