PCA_LDA_QDA_NB_FS

December 18, 2021

[3]:
```python
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib as plt


import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

[4]:
```python
app_df = pd.read_csv("archive/application_data.csv")
```

[5]:
```python
#missing data
missing_fractions = app_df.isnull().mean().sort_values(ascending=False)
missing_fractions.head(10)
```

[5]:
```
COMMONAREA_MEDI             0.698723
COMMONAREA_AVG              0.698723
COMMONAREA_MODE             0.698723
NONLIVINGAPARTMENTS_MODE    0.694330
NONLIVINGAPARTMENTS_AVG     0.694330
NONLIVINGAPARTMENTS_MEDI    0.694330
FONDKAPREMONT_MODE          0.683862
LIVINGAPARTMENTS_MODE       0.683550
LIVINGAPARTMENTS_AVG        0.683550
LIVINGAPARTMENTS_MEDI       0.683550
dtype: float64
```

1 1.Drop features

2 Limit the Feature Space

The full dataset has 122 features for each loan. We'll select features in two steps:

1. Drop features with more than 30% of their data missing.

2. Of the remaining features, choose only those that would be available to an investor before deciding to fund the loan.

[6]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
%matplotlib inline
mpl.style.use('ggplot')
sns.set(style='whitegrid')
```

3 1.1 Drop features missing more than 30% percent data

[7]:
```
drop_list = sorted(list(missing_fractions[missing_fractions > 0.3].index))
app_df.drop(labels=drop_list, axis=1, inplace=True)
```

4 Columns of choice

[8]:
```
#useless columns:
["SK_ID_CURR"]
app_df.drop(labels=["SK_ID_CURR"], axis=1, inplace=True)
```

5 Pearson correlation matrix

[9]:
```
# indicator (dummy variable) whether the applicant provided ...
Flag=['FLAG_MOBIL',
       'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE',
       'FLAG_EMAIL','FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3',
       'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6',
       'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9',
       'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12',
       'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15',
       'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18',
       'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21']
```

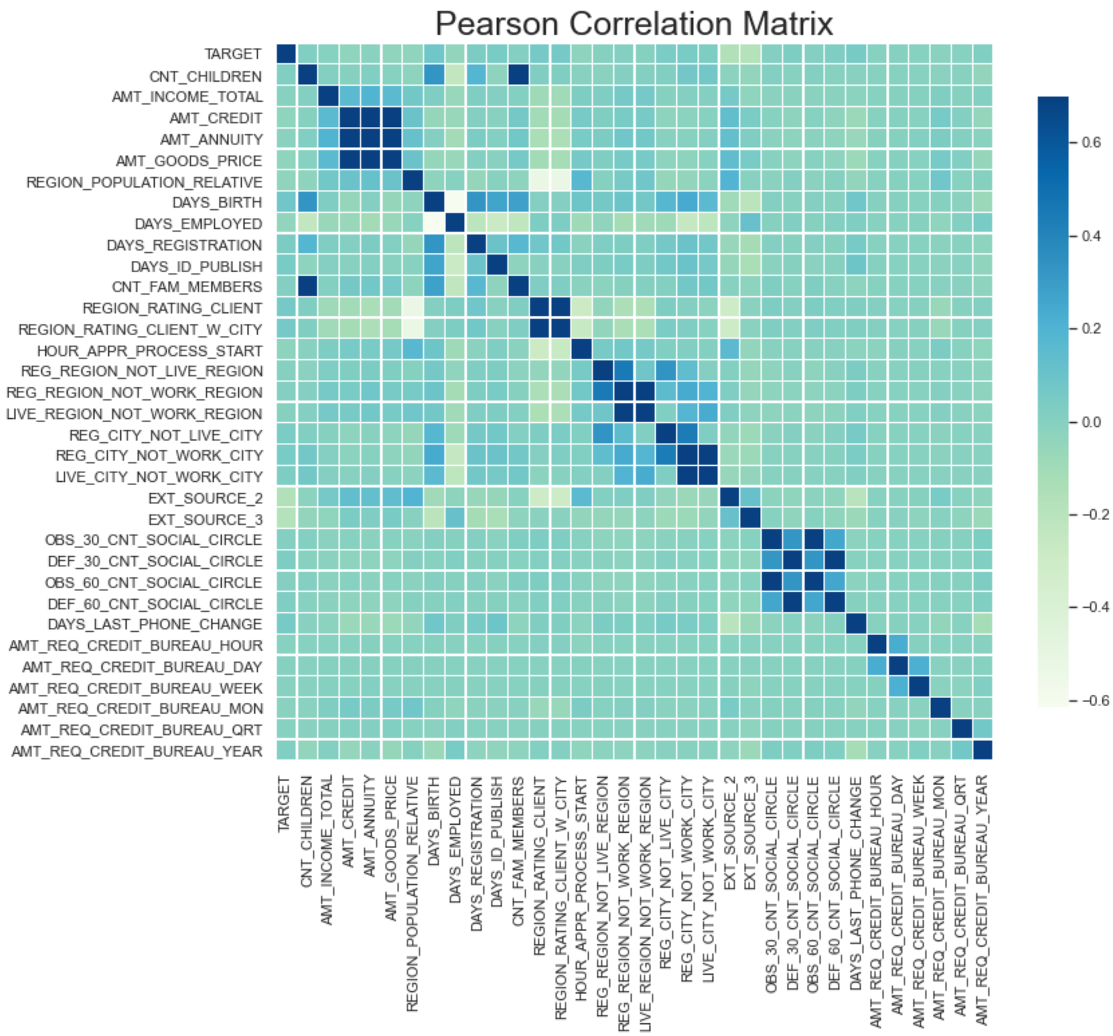
[10]:
```
app_df_2=app_df.drop(labels=Flag, axis=1)
```

[11]:
```
sns.set(style="whitegrid", font_scale=1)
plt.figure(figsize=(12,12))
plt.title('Pearson Correlation Matrix',fontsize=25)
sns.heatmap(app_df_2.corr(),linewidths=0.25,vmax=0.
 ↪7,square=True,cmap="GnBu",linecolor='w',
```

```python
                annot=False, cbar_kws={"shrink": .7})
```

[11]:

```
<AxesSubplot:title={'center':'Pearson Correlation Matrix'}>
```



[12]:

```python
drop_list_2=['AMT_ANNUITY','AMT_GOODS_PRICE','REGION_RATING_CLIENT']
```

[13]:

```python
app_df.drop(labels=drop_list_2, axis=1, inplace=True)
```

[14]:

```python
app_df
```

[14]:

```
   TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY  \
0       1         Cash loans           M            N               Y
1       0         Cash loans           F            N               N
2       0    Revolving loans           M            Y               Y
3       0         Cash loans           F            N               Y
```

```
4           0         Cash loans           M            N               Y
…        …                 …         …           …             …
307506      0         Cash loans           M            N               N
307507      0         Cash loans           F            N               Y
307508      0         Cash loans           F            N               Y
307509      1         Cash loans           F            N               Y
307510      0         Cash loans           F            N               N

        CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT NAME_TYPE_SUITE  \
0                  0          202500.0    406597.5   Unaccompanied
1                  0          270000.0   1293502.5          Family
2                  0           67500.0    135000.0   Unaccompanied
3                  0          135000.0    312682.5   Unaccompanied
4                  0          121500.0    513000.0   Unaccompanied
…                …                 …          …              …
307506             0          157500.0    254700.0   Unaccompanied
307507             0           72000.0    269550.0   Unaccompanied
307508             0          153000.0    677664.0   Unaccompanied
307509             0          171000.0    370107.0   Unaccompanied
307510             0          157500.0    675000.0   Unaccompanied

            NAME_INCOME_TYPE  … FLAG_DOCUMENT_18 FLAG_DOCUMENT_19  \
0                    Working  …                0                0
1              State servant  …                0                0
2                    Working  …                0                0
3                    Working  …                0                0
4                    Working  …                0                0
…                         …  …              …                …
307506               Working  …                0                0
307507             Pensioner  …                0                0
307508               Working  …                0                0
307509  Commercial associate  …                0                0
307510  Commercial associate  …                0                0

       FLAG_DOCUMENT_20  FLAG_DOCUMENT_21  AMT_REQ_CREDIT_BUREAU_HOUR  \
0                     0                 0                         0.0
1                     0                 0                         0.0
2                     0                 0                         0.0
3                     0                 0                         NaN
4                     0                 0                         0.0
…                   …               …                         …
307506                0                 0                         NaN
307507                0                 0                         NaN
307508                0                 0                         1.0
307509                0                 0                         0.0
307510                0                 0                         0.0
```

```
        AMT_REQ_CREDIT_BUREAU_DAY  AMT_REQ_CREDIT_BUREAU_WEEK  \
0                             0.0                         0.0
1                             0.0                         0.0
2                             0.0                         0.0
3                             NaN                         NaN
4                             0.0                         0.0
…                          …                          …
307506                        NaN                         NaN
307507                        NaN                         NaN
307508                        0.0                         0.0
307509                        0.0                         0.0
307510                        0.0                         0.0

        AMT_REQ_CREDIT_BUREAU_MON  AMT_REQ_CREDIT_BUREAU_QRT  \
0                             0.0                        0.0
1                             0.0                        0.0
2                             0.0                        0.0
3                             NaN                        NaN
4                             0.0                        0.0
…                          …                         …
307506                        NaN                        NaN
307507                        NaN                        NaN
307508                        1.0                        0.0
307509                        0.0                        0.0
307510                        2.0                        0.0

        AMT_REQ_CREDIT_BUREAU_YEAR
0                              1.0
1                              0.0
2                              0.0
3                              NaN
4                              0.0
…                           …
307506                         NaN
307507                         NaN
307508                         1.0
307509                         0.0
307510                         1.0

[307511 rows x 68 columns]
```

6 For linear model only, other team member please delete this part(2. Multicollinearity) in your code!

7 2. Multicollinearity

Although highly correlated features (multicollinearity) aren't a problem for the machine learning models based on decision trees (as used here), these features decrease importances of each other and can make feature analysis more difficult. Therefore, I calculate feature correlations and remove the features with very high correlation coefficients before applying machine learning.

[15]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

[16]:
```
#column that doesn't contain object input
value_column=[]
for i in range(app_df.shape[1]):
    if type(app_df.iloc[1,i])!=str:
        value_column.append(i)
```

[17]:
```
app_df_3=app_df.iloc[:,value_column]
```

[16]:
```
# Create a dataframe that will contain the names of all the feature variables␣
↪and their respective VIFs
vif = pd.DataFrame()
X=app_df_3.drop(labels=['TARGET'],axis=1)
#use the following line if you don't want to see the warning
#X=app_df_3.drop(labels=['FLAG_DOCUMENT_2'],axis=1).
↪drop(labels=['TARGET'],axis=1)
X=X.dropna()
vif['Features'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

```
/Users/yihomhguo/opt/anaconda3/lib/python3.7/site-
packages/statsmodels/regression/linear_model.py:1715: RuntimeWarning: invalid
value encountered in double_scalars
  return 1 - self.ssr/self.centered_tss
```

```
      ␣
↪---------------------------------------------------------------------------

       KeyboardInterrupt                         Traceback (most recent call␣
↪last)

       <ipython-input-16-c163844f1093> in <module>
```

```
      6 X=X.dropna()
      7 vif['Features'] = X.columns
----> 8 vif['VIF'] = [variance_inflation_factor(X.values, i) for i in␣
↪range(X.shape[1])]
      9 vif['VIF'] = round(vif['VIF'], 2)
     10 vif = vif.sort_values(by = "VIF", ascending = False)


    <ipython-input-16-c163844f1093> in <listcomp>(.0)
      6 X=X.dropna()
      7 vif['Features'] = X.columns
----> 8 vif['VIF'] = [variance_inflation_factor(X.values, i) for i in␣
↪range(X.shape[1])]
      9 vif['VIF'] = round(vif['VIF'], 2)
     10 vif = vif.sort_values(by = "VIF", ascending = False)


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/stats/
↪outliers_influence.py in variance_inflation_factor(exog, exog_idx)
    190     mask = np.arange(k_vars) != exog_idx
    191     x_noti = exog[:, mask]
--> 192     r_squared_i = OLS(x_i, x_noti).fit().rsquared
    193     vif = 1. / (1. - r_squared_i)
    194     return vif


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/regression/
↪linear_model.py in fit(self, method, cov_type, cov_kwds, use_t, **kwargs)
    303                     hasattr(self, 'rank')):
    304
--> 305                 self.pinv_wexog, singular_values =␣
↪pinv_extended(self.wexog)
    306                 self.normalized_cov_params = np.dot(
    307                     self.pinv_wexog, np.transpose(self.pinv_wexog))


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/tools/tools.py␣
↪in pinv_extended(x, rcond)
    405     x = np.asarray(x)
    406     x = x.conjugate()
--> 407     u, s, vt = np.linalg.svd(x, False)
    408     s_orig = np.copy(s)
    409     m = u.shape[0]


    <__array_function__ internals> in svd(*args, **kwargs)
```

```
    ~/opt/anaconda3/lib/python3.7/site-packages/numpy/linalg/linalg.py in␣
↪svd(a, full_matrices, compute_uv, hermitian)
   1659
   1660         signature = 'D->DdD' if isComplexType(t) else 'd->ddd'
-> 1661         u, s, vh = gufunc(a, signature=signature, extobj=extobj)
   1662         u = u.astype(result_t, copy=False)
   1663         s = s.astype(_realType(result_t), copy=False)


KeyboardInterrupt:
```

[18]:
```
app_df_3=app_df_3.
 ↪drop(labels=['FLAG_MOBIL','FLAG_EMP_PHONE','OBS_60_CNT_SOCIAL_CIRCLE'],␣
 ↪axis=1)
```

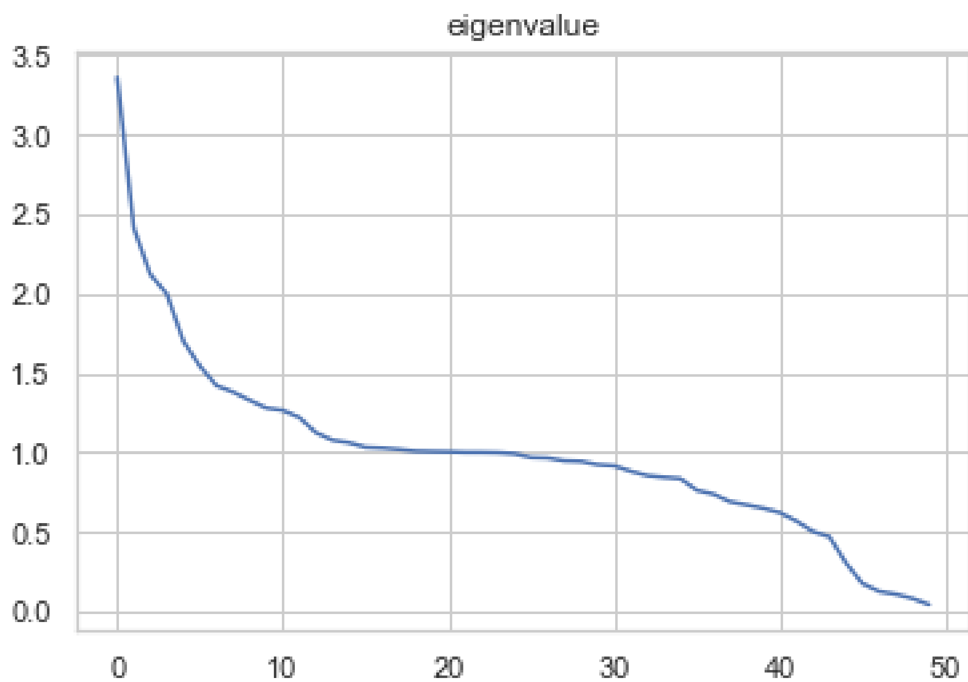
8 3. Your code

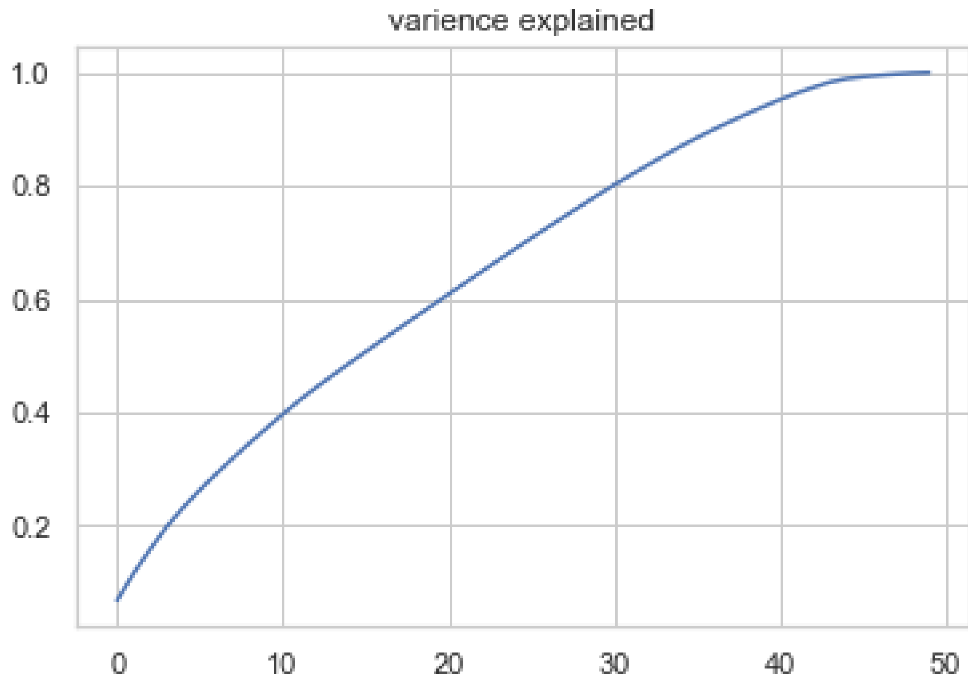
[19]:
```
#  one-hot encoding is not necessary for some models! Please be aware.
app_df_one_hot=pd.get_dummies(app_df_3)
```

[20]:
```
from sklearn.model_selection import train_test_split
from imblearn.under_sampling import RandomUnderSampler



app_df_one_hot = app_df_one_hot.dropna(axis=0)
X = app_df_one_hot.drop(['TARGET'],axis=1)
y = app_df_one_hot['TARGET']


x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.3)

rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(x_train, y_train)
```

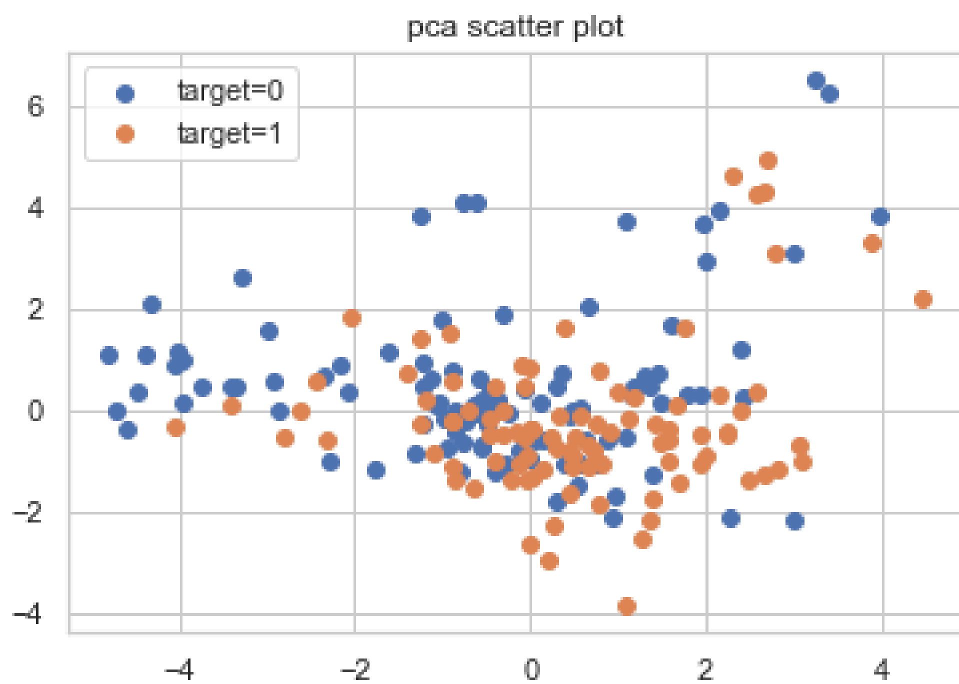
[106]:
```
from sklearn.decomposition import PCA
from matplotlib.colors import ListedColormap


xx = ((X_resampled-np.mean(X_resampled,axis=0))/np.std(X_resampled,axis=0)).
 ↪dropna(axis=1)
n_components = np.array(xx).shape[1]
```

```
pca = PCA(n_components=n_components)
X_pca = pca.fit_transform(xx)
L = pca.explained_variance_
plt.plot(L);
plt.title('eigenvalue')
plt.show()
# print("Variance explained principal components:",L)
plt.plot(np.arange(len(L)),np.cumsum(L)/sum(L))
plt.title('varience explained')

plt.show()
cmap_bold = ListedColormap(['red', 'blue'])
plot_index = np.random.randint(len(xx),size=200)
plt.scatter(X_pca[plot_index][np.array(y_resampled)[plot_index]==0,0],␣
 ↪X_pca[plot_index][np.array(y_resampled)[plot_index]==0,1])
plt.scatter(X_pca[plot_index][np.array(y_resampled)[plot_index]==1,0],␣
 ↪X_pca[plot_index][np.array(y_resampled)[plot_index]==1,1])
plt.title('pca scatter plot')
plt.legend(['target=0','target=1'])
```



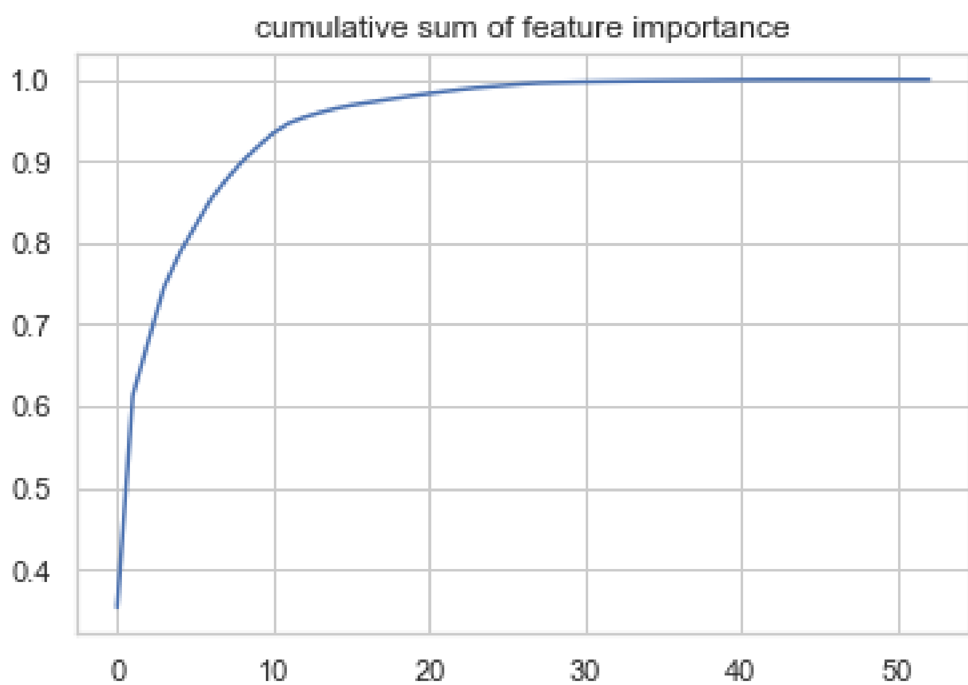


[106]: <matplotlib.legend.Legend at 0x7fdcf496b390>



[107]:
```
# feature selection
from sklearn.ensemble import RandomForestClassifier
clf_rf=RandomForestClassifier(max_depth=5,n_estimators=100)
clf_rf.fit(X_resampled,y_resampled)


clf_rf.feature_importances_
X_resampled.columns
imp = {}
for i in range(len(X_resampled.columns)):
    imp[X_resampled.columns[i]] = clf_rf.feature_importances_[i]

d_order=sorted(imp.items(),key=lambda x:x[1],reverse=True)
key = [i[0] for i in d_order]
value = [i[1] for i in d_order]
plt.plot(np.arange(len(value)),np.cumsum(value))
plt.title('cumulative sum of feature importance')
```

[107]:
```
Text(0.5, 1.0, 'cumulative sum of feature importance')
```



[75]:
```
# pick first ten feature
feature = key[:10]
```

experiment on all feature, first ten feature and first two feature based on random forest feature importance.

Perform Naive Bayes, LDA and QDA

Performance evaluated on AUC and accuracy

[86]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA

LDA_model = LinearDiscriminantAnalysis()
LDA_model.fit(X_resampled[['EXT_SOURCE_2', 'EXT_SOURCE_3']],y_resampled)

y_pred = LDA_model.predict_proba(x_test[['EXT_SOURCE_2', 'EXT_SOURCE_3']])[:,1]
y_pred_cls=  LDA_model.predict(x_test[['EXT_SOURCE_2', 'EXT_SOURCE_3']])
print('train')
print('accuracy','auc')
y_pred = LDA_model.predict_proba(X_resampled[['EXT_SOURCE_2',␣
 ↪'EXT_SOURCE_3']])[:,1]
y_pred_cls=  LDA_model.predict(X_resampled[['EXT_SOURCE_2', 'EXT_SOURCE_3']])
print(sum(y_pred_cls==y_resampled)/len(X_resampled))
print(auc(y_resampled,y_pred))


print('test')
y_pred = LDA_model.predict_proba(x_test[['EXT_SOURCE_2', 'EXT_SOURCE_3']])[:,1]
y_pred_cls=  LDA_model.predict(x_test[['EXT_SOURCE_2', 'EXT_SOURCE_3']])
print(sum(y_pred_cls==y_test)/len(y_test))
print(auc(y_test,y_pred))
```

```
train
0.6670431443415406
0.7226859635725973
test
0.6778606458273899
0.7165614137592675
```

[94]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA

LDA_model = LinearDiscriminantAnalysis()
LDA_model.fit(X_resampled[feature],y_resampled)
print('train')
print('accuracy','auc')
y_pred = LDA_model.predict_proba(X_resampled[feature])[:,1]
y_pred_cls=  LDA_model.predict(X_resampled[feature])
print(sum(y_pred_cls==y_resampled)/len(X_resampled))
print(auc(y_resampled,y_pred))

print('test')
y_pred = LDA_model.predict_proba(x_test[feature])[:,1]
```

```
y_pred_cls=  LDA_model.predict(x_test[feature])
print(sum(y_pred_cls==y_test)/len(y_test))
print(auc(y_test,y_pred))
```

```
train
0.6744597545365559
0.7308516512357309
test
0.6799391395307767
0.7237661161448519
```

[95]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA

LDA_model = LinearDiscriminantAnalysis()
LDA_model.fit(X_resampled,y_resampled)

print('train')
print('accuracy','auc')
y_pred = LDA_model.predict_proba(X_resampled)[:,1]
y_pred_cls=  LDA_model.predict(X_resampled)
print(sum(y_pred_cls==y_resampled)/len(X_resampled))
print(auc(y_resampled,y_pred))

print('test')
y_pred = LDA_model.predict_proba(x_test)[:,1]
y_pred_cls=  LDA_model.predict(x_test)
print(sum(y_pred_cls==y_test)/len(y_test))
print(auc(y_test,y_pred))
```

```
train
0.6769445071907236
0.7376274443261633
test
0.6822485769789841
0.7303508263416456
```

[87]:
```
LDA_model = QDA()
LDA_model.fit(X_resampled[['EXT_SOURCE_2', 'EXT_SOURCE_3']],y_resampled)
print('train')
print('accuracy','auc')

y_pred = LDA_model.predict_proba(X_resampled[['EXT_SOURCE_2',␣
 ↪'EXT_SOURCE_3']])[:,1]
y_pred_cls=  LDA_model.predict(X_resampled[['EXT_SOURCE_2', 'EXT_SOURCE_3']])
print(sum(y_pred_cls==y_resampled)/len(X_resampled))
print(auc(y_resampled,y_pred))
```

```
print('test')
y_pred = LDA_model.predict_proba(x_test[['EXT_SOURCE_2', 'EXT_SOURCE_3']])[:,1]
y_pred_cls=  LDA_model.predict(x_test[['EXT_SOURCE_2', 'EXT_SOURCE_3']])
print(sum(y_pred_cls==y_test)/len(y_test))
print(auc(y_test,y_pred))
```

```
train
0.662525412243054
0.7198274148780746
test
0.6862968849764302
0.7134886253869688
```

[93]:

```
LDA_model = QDA()
LDA_model.fit(X_resampled[feature],y_resampled)
print('train')
print('accuracy','auc')

y_pred = LDA_model.predict_proba(X_resampled[feature])[:,1]
y_pred_cls=  LDA_model.predict(X_resampled[feature])
print(sum(y_pred_cls==y_resampled)/len(X_resampled))
print(auc(y_resampled,y_pred))

print('test')
y_pred = LDA_model.predict_proba(x_test[feature])[:,1]
y_pred_cls=  LDA_model.predict(x_test[feature])
print(sum(y_pred_cls==y_test)/len(y_test))
print(auc(y_test,y_pred))
```

```
train
0.6646713349898351
0.7206683227375603
test
0.6556221216937685
0.7109903988891854
```

[92]:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA

LDA_model = QDA()
LDA_model.fit(X_resampled,y_resampled)
print('train')
print('accuracy','auc')

y_pred = LDA_model.predict_proba(X_resampled)[:,1]
```

```
y_pred_cls=  LDA_model.predict(X_resampled)
print(sum(y_pred_cls==y_resampled)/len(X_resampled))
print(auc(y_resampled,y_pred))

print('test')
y_pred = LDA_model.predict_proba(x_test)[:,1]
y_pred_cls=  LDA_model.predict(x_test)
print(sum(y_pred_cls==y_test)/len(y_test))
print(auc(y_test,y_pred))
```

```
/Users/yihomhguo/opt/anaconda3/lib/python3.7/site-
packages/sklearn/discriminant_analysis.py:878: UserWarning: Variables are
collinear
  warnings.warn("Variables are collinear")
```

```
train
0.5055718695881334
0.5174095879976979
test
0.0936952357663936
0.5144287117655393
```

[91]:
```
from sklearn.naive_bayes import GaussianNB

LDA_model = GaussianNB()
LDA_model.fit(X_resampled,y_resampled)
print('train')
print('accuracy','auc')

y_pred = LDA_model.predict_proba(X_resampled)[:,1]
y_pred_cls=  LDA_model.predict(X_resampled)
print(sum(y_pred_cls==y_resampled)/len(X_resampled))
print(auc(y_resampled,y_pred))

print('test')
y_pred = LDA_model.predict_proba(x_test)[:,1]
y_pred_cls=  LDA_model.predict(x_test)
print(sum(y_pred_cls==y_test)/len(y_test))
print(auc(y_test,y_pred))
```

```
train
0.57811911753633
0.613450973294713
test
0.44945728219967124
0.6105408669202502
```

[90]:

```python
from sklearn.naive_bayes import GaussianNB

LDA_model = GaussianNB()
LDA_model.fit(X_resampled[feature],y_resampled)

print('train')
print('accuracy','auc')

y_pred = LDA_model.predict_proba(X_resampled[feature])[:,1]
y_pred_cls=  LDA_model.predict(X_resampled[feature])
print(sum(y_pred_cls==y_resampled)/len(X_resampled))
print(auc(y_resampled,y_pred))

print('test')
y_pred = LDA_model.predict_proba(x_test[feature])[:,1]
y_pred_cls=  LDA_model.predict(x_test[feature])
print(sum(y_pred_cls==y_test)/len(y_test))
print(auc(y_test,y_pred))
```

```
train
0.5810932911678337
0.6138853413925691
test
0.450978793930255
0.6092950849821512
```

[89]:

```python
from sklearn.naive_bayes import GaussianNB

LDA_model = GaussianNB()
LDA_model.fit(X_resampled[['EXT_SOURCE_2', 'EXT_SOURCE_3']],y_resampled)
print('train')
print('accuracy','auc')

y_pred = LDA_model.predict_proba(X_resampled[['EXT_SOURCE_2',␣
 ↪'EXT_SOURCE_3']])[:,1]
y_pred_cls=  LDA_model.predict(X_resampled[['EXT_SOURCE_2', 'EXT_SOURCE_3']])
print(sum(y_pred_cls==y_resampled)/len(X_resampled))
print(auc(y_resampled,y_pred))


print('test')
y_pred = LDA_model.predict_proba(x_test[['EXT_SOURCE_2', 'EXT_SOURCE_3']])[:,1]
y_pred_cls=  LDA_model.predict(x_test[['EXT_SOURCE_2', 'EXT_SOURCE_3']])
print(sum(y_pred_cls==y_test)/len(y_test))
print(auc(y_test,y_pred))
```

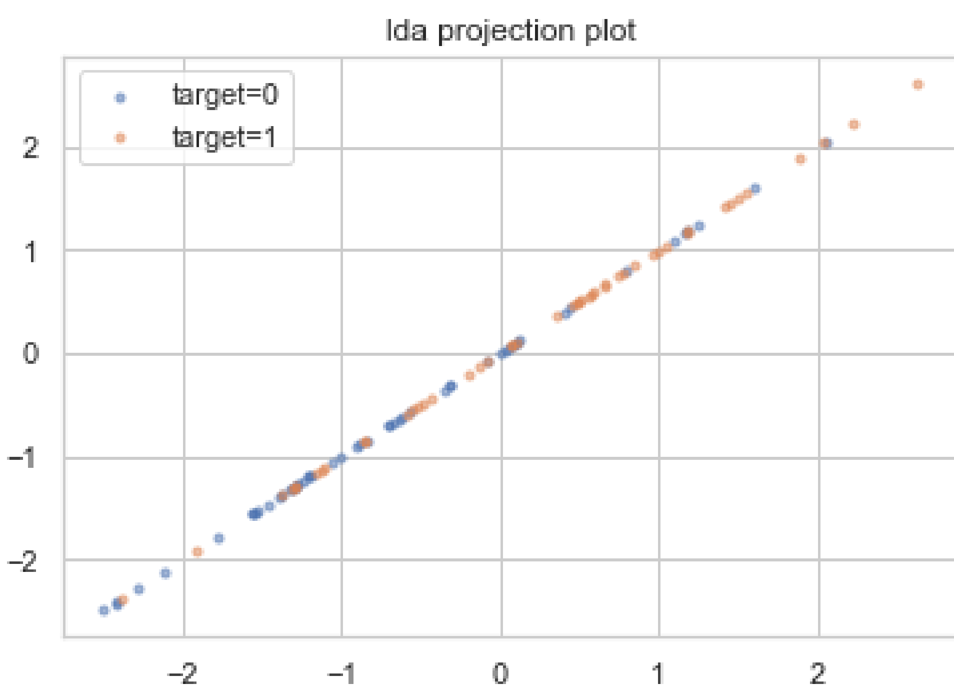
```
train
```

```
0.6622618778706423
0.7205541208243939
test
0.6926274605697518
0.71418064121375
```

[ ]:

```
[105]: from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA

LDA_model = LinearDiscriminantAnalysis(n_components=1)
LDA_model.fit(X_resampled,y_resampled)
X_pca = LDA_model.transform(X_resampled)

plot_index = np.random.randint(len(xx),size=100)
plt.scatter(X_pca[plot_index][np.array(y_resampled)[plot_index]==0,0],␣
 ↪X_pca[plot_index][np.array(y_resampled)[plot_index]==0,0],alpha=0.5,s=10)
plt.scatter(X_pca[plot_index][np.array(y_resampled)[plot_index]==1,0],␣
 ↪X_pca[plot_index][np.array(y_resampled)[plot_index]==1,0],alpha=0.5,s=10)
plt.legend(['target=0','target=1'])
plt.title('lda projection plot')
```

```
[105]: Text(0.5, 1.0, 'lda projection plot')
```



[ ]:

```

```