discriminant_analysis_1

September 30, 2019

[1]:
```
# src

# https://medium.com/journey-2-artificial-intelligence/
↪lda-linear-discriminant-analysis-using-python-2155cf5b6398

# https://github.com/sambit9238/DataScience/blob/master/LDA.ipynb?
↪source=post_page-----2155cf5b6398---------------------
```

[2]:
```
# LDA is a supervised dimensionality reduction technique
# The goal is to project a dataset onto a lower-dimensional space with good␣
↪class-separability in order avoid overfitting (curse of dimensionality) and␣
↪also reduce computational costs

# Basically, the added advantage LDA gives over PCA is to tackle overfitting.

# The general LDA approach is very similar to a Principal Component Analysis.
# But in addition to finding the component axes that maximize the variance of␣
↪our data (PCA), we are additionally interested in the axes that maximize the␣
↪separation between multiple classes (LDA).

# Steps of LDA:
# Compute d-dimensional mean vectors for different classes from the dataset,␣
↪where d is the dimension of feature space.
# Compute in-between class and with-in class scatter matrices.
# Compute eigen vectors and corresponding eigen values for the scatter matrices.
# Choose k eigen vectors corresponding to top k eigen values to form a␣
↪transformation matrix of dimension d x k.
# Transform the d-dimensional feature space X to k-dimensional feature space␣
↪X_lda via the transformation matrix.
```

[3]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

import numpy as np

import pandas as pd
```

```python
import scipy.stats as sstats
```

[4]:
```python
CSV_PATH = 'http://archive.ics.uci.edu/ml/machine-learning-databases/00267/
↪data_banknote_authentication.txt'
main_df = pd.read_csv(
    CSV_PATH,
    names=['var', 'skewness', 'curtosis', 'entropy', 'class'],
    index_col=False
)
```

[5]:
```python
display( main_df.shape )
display( main_df.sample() )
display( main_df.isnull().sum() )
display( main_df.duplicated().sum() )

display( main_df['class'].value_counts() )

display( main_df.describe(include='all') )

main_df.info()
```

```
(1372, 5)
```

	var	skewness	curtosis	entropy	class
400	1.3049	-0.15521	6.4911	-0.75346	0

```
var         0
skewness    0
curtosis    0
entropy     0
class       0
dtype: int64
```

```
24
```

```
0    762
1    610
Name: class, dtype: int64
```

	var	skewness	curtosis	entropy	class
count	1372.000000	1372.000000	1372.000000	1372.000000	1372.000000
mean	0.433735	1.922353	1.397627	-1.191657	0.444606
std	2.842763	5.869047	4.310030	2.101013	0.497103

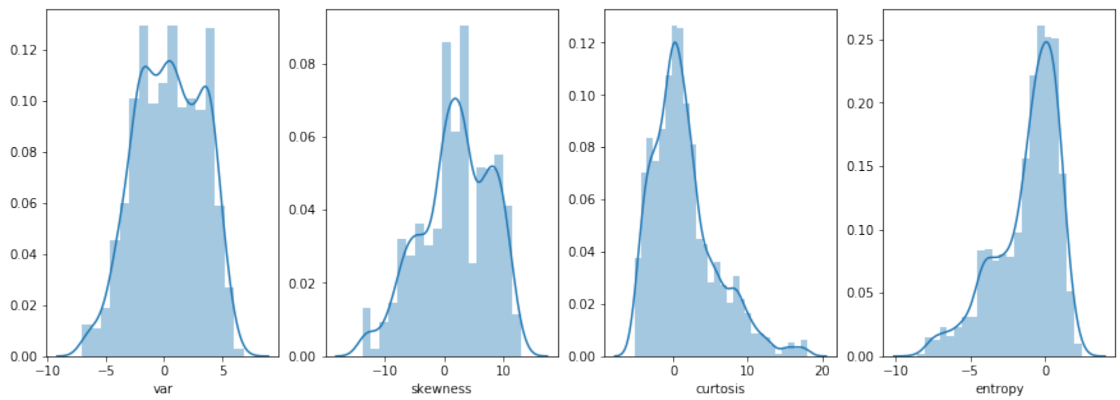
```
min      -7.042100   -13.773100   -5.286100   -8.548200    0.000000
25%      -1.773000    -1.708200   -1.574975   -2.413450    0.000000
50%       0.496180     2.319650    0.616630   -0.586650    0.000000
75%       2.821475     6.814625    3.179250    0.394810    1.000000
max       6.824800    12.951600   17.927400    2.449500    1.000000
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1372 entries, 0 to 1371
Data columns (total 5 columns):
var          1372 non-null float64
skewness     1372 non-null float64
curtosis     1372 non-null float64
entropy      1372 non-null float64
class        1372 non-null int64
dtypes: float64(4), int64(1)
memory usage: 53.7 KB
```

[6]:

```
# If the K-S statistic is small or the p-value is high, then
# we cannot reject the hypothesis that the distributions of the two samples
# are the same.

fig, ax = plt.subplots( 1, 4, figsize=(15, 5) )
for idx, col_name in enumerate( main_df.columns[:-1] ):  # skip 'class' column
    sns.distplot( main_df[col_name], ax=ax[idx] )
    print('{0}: skew:{1}, kurt:{2}, KS-score:{3}\n'.format(
        col_name,
        main_df[col_name].skew(),
        main_df[col_name].kurt(),
        sstats.kstest( main_df[col_name], 'norm' )
    ))
plt.show()
```

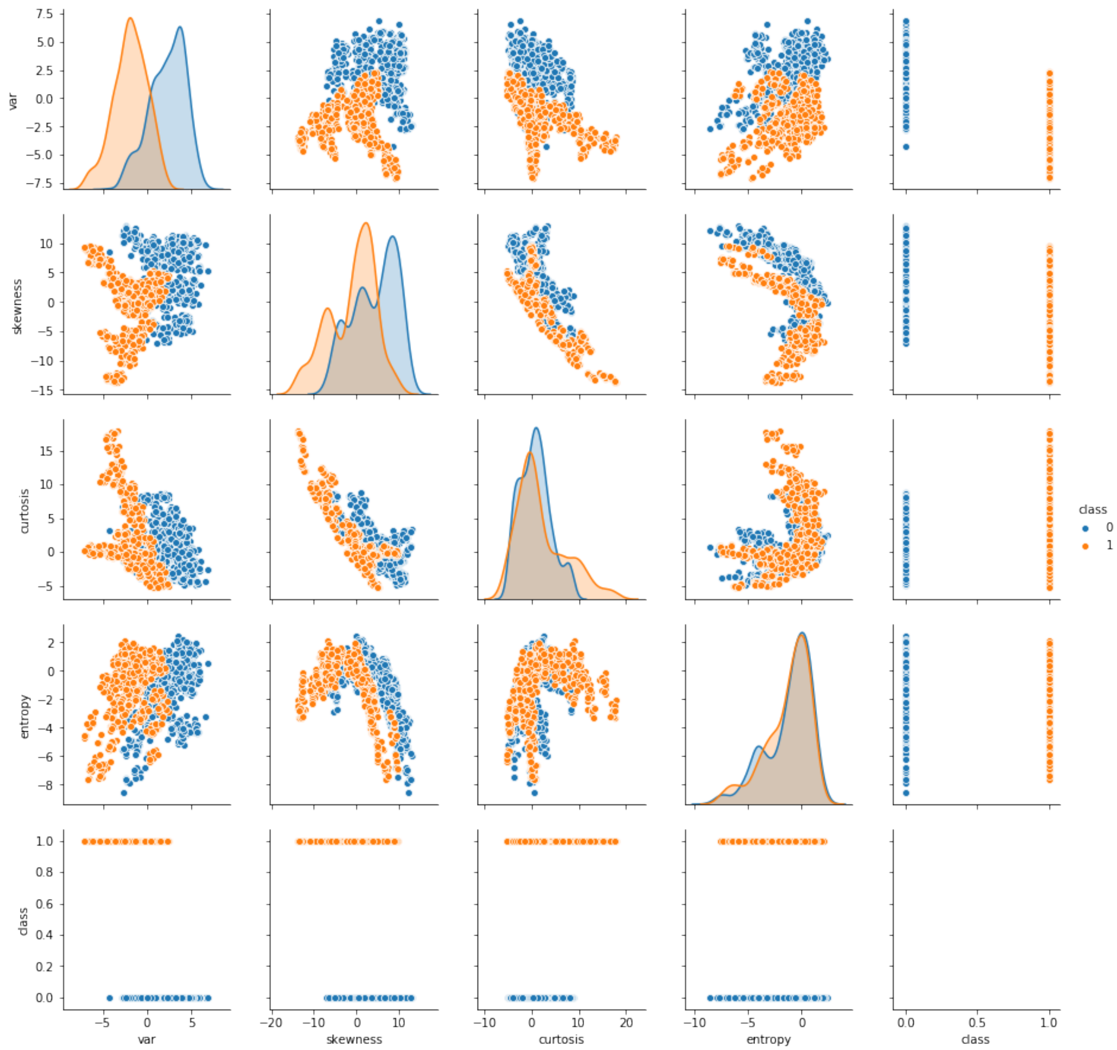
```
var: skew:-0.14938770055109987, kurt:-0.7515813815834465, KS-
score:KstestResult(statistic=0.3178576100116328, pvalue=9.218224874530454e-124)

skewness: skew:-0.39410347444624066, kurt:-0.437211524327382, KS-
score:KstestResult(statistic=0.49891659792840465, pvalue=3.16694794e-316)

curtosis: skew:1.088568543275335, kurt:1.2704759157901702, KS-
score:KstestResult(statistic=0.3373300744120429, pvalue=9.162592642100847e-140)

entropy: skew:-1.0222430438083978, kurt:0.49749575397598766, KS-
score:KstestResult(statistic=0.27077469676882676, pvalue=2.3923266260137977e-89)
```



[7]:
```
sns.pairplot(
    main_df,
    hue='class'
)
```

```
/home/max/.conda/envs/studyingenv/lib/python3.7/site-
packages/statsmodels/nonparametric/kde.py:487: RuntimeWarning: invalid value
encountered in true_divide
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
/home/max/.conda/envs/studyingenv/lib/python3.7/site-
packages/statsmodels/nonparametric/kdetools.py:34: RuntimeWarning: invalid value
encountered in double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
```

[7]: <seaborn.axisgrid.PairGrid at 0x7fb319788da0>



[8]:
```
# Compute the 4-dimensional mean vectors for both the classes
# Unlike PCA, standardization of the data is not needed in LDA as it doesn't
 ↪affect the output.

mean_vec = []

for unique_class_value in main_df['class'].unique():
    mean_vec.append(
        np.array( main_df[ main_df['class'] == 0 ].mean()[:4] )
    )
```

[9]:
```
# Calculate:
# 1. the with-in class scatter matrices
# 2. in-between class scatter matrices

# With-in class scatter matrices
```

```python
SW = np.zeros( (4, 4) )

for i in range(2):  # for each unique class value
    per_class_sc_mat = np.zeros( (4, 4) )
    for j in range( main_df[main_df["class"] == i].shape[0] ):
        row = main_df.loc[j][:4].values.reshape(4,1)
        mv = mean_vec[i].reshape(4,1)
        per_class_sc_mat += (row - mv).dot( (row - mv).T )
    SW += per_class_sc_mat


# In-between class scatter matrices
overall_mean = np.array(main_df.drop("class", axis=1).mean())

SB = np.zeros( (4, 4) )

for i in range(2):
    n = main_df[main_df["class"]==i].shape[0]
    mv = mean_vec[i].reshape(4,1)
    overall_mean = overall_mean.reshape(4,1) # make column vector
    SB += n * (mv - overall_mean).dot((mv - overall_mean).T)
```

[10]:
```python
# Solve the generalized eigenvalue problem to obtain the linear discrirminants

e_vals, e_vecs = np.linalg.eig(  # eigenvalues and eigenvectors
    np.linalg.inv( SW ).dot( SB )
)
```

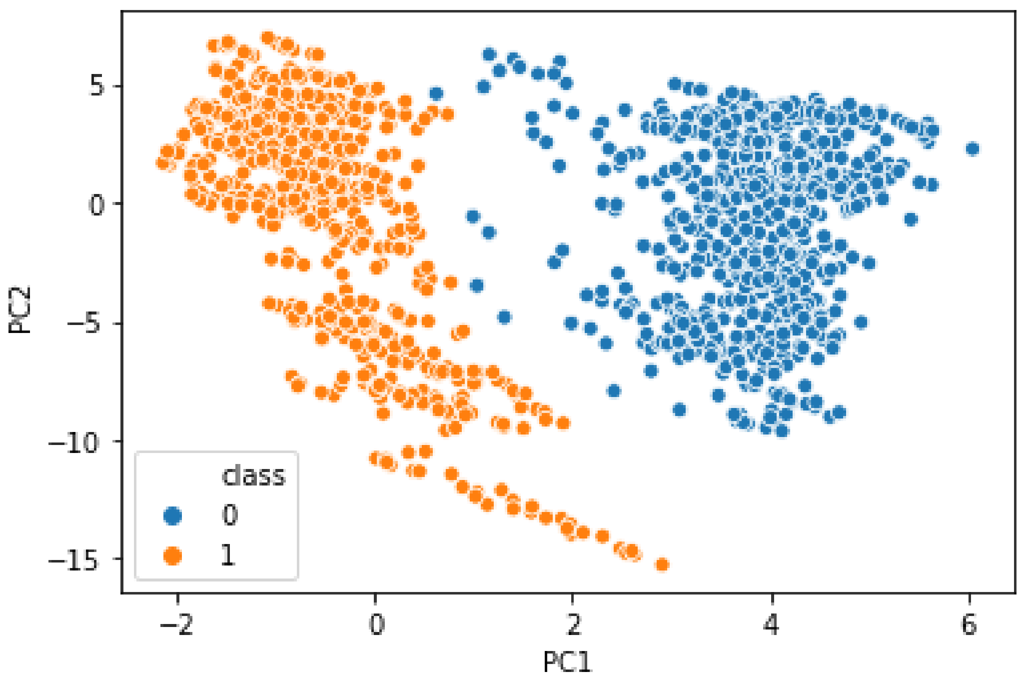
[11]:
```python
# Make a list of (eigenvalue, eigenvector) tuples
# Sort the tuples from high to low values

e_pairs = [
    ( np.abs(e_vals[i]), e_vecs[:,i] )
    for i in range( len(e_vals) )
]

e_pairs.sort( reverse=True )
```

[12]:
```python
# Select top k eigenvectors corresponding to top k eigenvalues

# For data compression purpose, we generally go for 99% variance retention,␣
 ↪while for visualization we make the dimension to 2 or 3.

# Here, we till take top-2 eigen values corresponding eigen vectors for␣
 ↪visualization purpose.
# But we will the eigen vector belongs to largest eigen value retains nearly␣
 ↪100% variance, so we can discard other 3 too.
```

```
W = np.hstack((
    e_pairs[0][1].reshape(4,1),
    e_pairs[1][1].reshape(4,1)
))
```

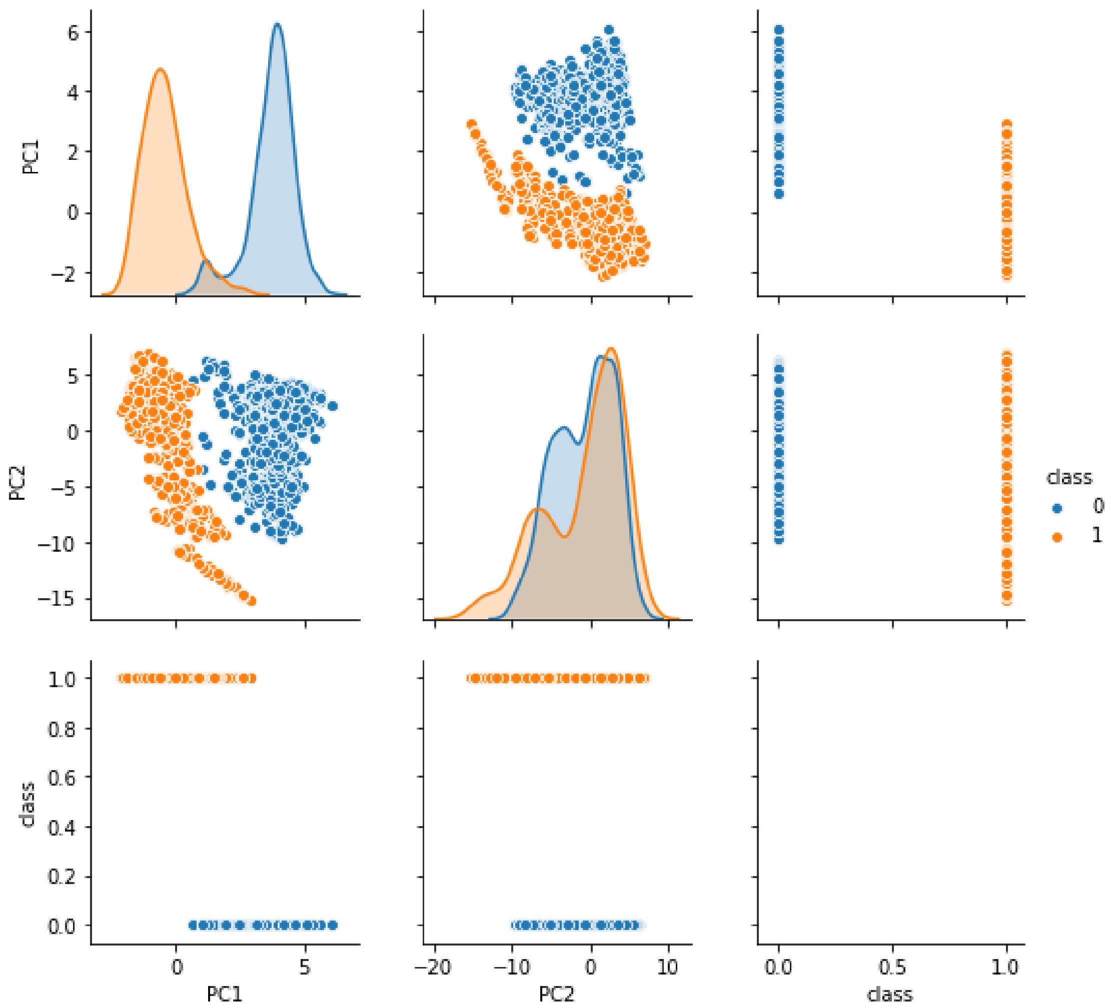
[13]:
```
# Transform the 4-dim feature space to 2-dim feature subspace

X = main_df.iloc[:,0:4].values
X_lda = X.dot(W)

main_df["PC1"] = X_lda[:,0]
main_df["PC2"] = X_lda[:,1]
```

[14]:
```
# Visualize 2 new components

sns.scatterplot(
    x='PC1', y='PC2',
    hue='class',
    data=main_df,
)
plt.show()
```



[15]:
```
sns.pairplot(
    main_df[ ['PC1', 'PC2', 'class'] ],
    hue='class'
)
```

[15]: <seaborn.axisgrid.PairGrid at 0x7fb2f293e940>



[16]:
```
# sklearn and LDA

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

X = main_df.iloc[:,0:4].values
y = main_df.iloc[:,4].values

model_lda = LDA( n_components=2 )

X_lda_skl = model_lda.fit_transform( X, y )
main_df['skl_PC1'] = X_lda_skl[:, 0]
```

```
/home/max/.conda/envs/studyingenv/lib/python3.7/site-
packages/sklearn/discriminant_analysis.py:466: ChangedBehaviorWarning:
```

```
n_components cannot be larger than min(n_features, n_classes - 1). Using
min(n_features, n_classes - 1) = min(4, 2 - 1) = 1 components.
  ChangedBehaviorWarning)
/home/max/.conda/envs/studyingenv/lib/python3.7/site-
packages/sklearn/discriminant_analysis.py:472: FutureWarning: In version 0.23,
setting n_components > min(n_features, n_classes - 1) will raise a ValueError.
You should set n_components to None (default), or a value smaller or equal to
min(n_features, n_classes - 1).
  warnings.warn(future_msg, FutureWarning)
```

[17]:
```
# Visualize results for LDA from sklearn

sns.scatterplot(
    x='skl_PC1', y='class',
    data=main_df
)
```

[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb2e945b048>
```