Xilinx Demo

Meher Krishna Patel

Created on : September, 2018

Last updated : October, 2018

Table of contents

Table of contents i

List of figures ii

List of tables iii

1 Machine learning Demo 1
- 1.1 Aim . . . 1
- 1.2 Basic information . . . 1
  - 1.2.1 Data analysis and Machine learning . . . 1
  - 1.2.2 Knowledge required . . . 2
  - 1.2.3 Classification . . . 2
- 1.3 Demo: Chronic Kidney Disease . . . 2
  - 1.3.1 How PCA works . . . 3
  - 1.3.2 Read and clean data . . . 3
  - 1.3.3 Remove categorical data . . . 4
  - 1.3.4 Apply PCA analysis . . . 6
  - 1.3.5 PCA limitation . . . 7
  - 1.3.6 Convert 'categorical' features to 'numeric' features . . . 10
- 1.4 Summary . . . 12

List of figures

1.1 Result of PCA analysis 8
1.2 Result of PCA analysis after preprocessing the data 10
1.3 Result of PCA analysis after converting categorical features to numerical feature 12

List of tables

1.1 Classification of Machine learning 2
1.2 Types of variable 2

Chapter 1

Machine learning Demo

1.1 Aim

- Understand the concept of data analysis and machine learning algorithm
- Difference between **designing** the ML algorithm and **using** the ML algorithm
- Classification of ML algorithms
- Demo to show the various steps of the data analysis process.

Note:

- `Click here` to download the final code of this demo.
- More demo examples and Python tutorials : https://pythondsp.readthedocs.io/en/latest/pythondsp/toc.html#python

---

1.2 Basic information

1.2.1 Data analysis and Machine learning

Machine learning algorithms are a part of data analysis process. The data analysis process involves the following steps,

1. Collecting the data from various sources.
2. Cleaning and rearranging the data e.g. filling the missing values from the dataset, removing is irrelevant data etc.
3. Exploring the data e.g. checking the statistical values of the data and visualizing the data using plots etc.
4. Modeling the data using correct machine learning algorithms (if required).
5. Lastly, check the performance of the newly created model.

Note:

- When we collect data, we collect everything which is available (without giving a thought). During collection, some of the information may not be available (or corrupted) for each sample. For example, in the user-information form, some of the people may not be willing to give their phone numbers.
- Next we need to clean the data according to the application. This is **80%** of the total data analysis process. For example, name of city has no relation to the temperature of the city (only location is important). Feeding the irrelevant data will result in improper learning, and the ML algorithm will give wrong results. Therefore, **domain knowledge** is an essential part of the data analysis process.

1.2.2 Knowledge required

Data analysis requires the knowledge of multiple fields e.g.

- Data cleaning using Python or R language i.e. filling the missing/corrupted data and filtering the irrelevant data etc.
- Good knowledge of mathematics for measuring the statistical parameter of the data (required for data cleaning).
- Knowledge of some specific field on which we want to apply the machine learning algorithm (helpful in creating dataset for the algorithm).
- Lastly, we must have the understanding of the machine learning algorithms.

Note:

1. Not all the problems can be solved using Machine learning algorithms.
2. If a problem can be solved directly, then do not use machine learning algorithms.
3. Each machine learning algorithms has it's own advantages and disadvantages. In the other words, we need to choose the correct machine learning algorithms to solve the problem.
4. We need not to be expert in the mathematics behind the machine learning algorithms; but we should be aware of pros and cons of the algorithms.
5. The sound knowledge of advance mathematics is required for designing the ML algorithms.

1.2.3 Classification

- Supervised learning : We have output samples (also called targets).
  - Classification : discrete outputs e.g. good/bad, male/female etc.
  - Regression : continuous outputs e.g. height, age etc.
- Unsupervised learning: We have only data and output samples.
  - Clustering : Try to make the groups based on the data e.g. make a cluster based on hobbies etc.
  - Dimensionality reduction : Remove the correlated data; e.g. temperature and weather (hot/cold) are correlated and one of these may be neglected based on application (or combined into one sample based on certain algorithms).

Note: Real world problem can be a combination of Supervised and Unsupervised learning i.e. first we need to find one of the samples as the output sample (i.e. Unsupervised) and then use other sample for ML algorithms (i.e. Supervised). Or we can use clustering with dimensionality reduction etc.

Table 1.1: Classification of Machine learning

Machine learning	Subtypes
Supervised	Binary classification, multiclass classification, regression
Unsupervised	Clustering, Dimensionality reduction

Table 1.2: Types of variable

Type	Description
categorical or factor	string (e.g. Male/Female), or fixed number of integers 0/1/2
numeric	floating point values

1.3 Demo: Chronic Kidney Disease

In this section we will use Principal component analysis (PCA) algorithms to detect the chronic kidney disease based on the blood sample.

Note: In the file "chronic_kidney_disease.arff" various tests are performed on the blood sample of the kidney patients. And our aim is to use the PCA algorithms to detect the possibility of kidney disease in the new patients.

The file can be downloaded from below link,

- http://archive.ics.uci.edu/ml/datasets/Chronic_Kidney_Disease

1.3.1 How PCA works

During the data collection process, our aim is to collect as much as data possible. During this process, it might possible some of the 'features' are correlated. If the dataset has lots of features, then it is good to remove some of the correlated features, so that the data can be processed faster; but at the same time the accuracy of the model may be reduced.

1.3.2 Read and clean data

First step is to read and clean the data. It is the one of the most challenging part in the Data analysis process as the dataset can be quite big (upto Tera-bytes) from various sources.

Listing 1.1: Read data from file and remove missing line

```
# kidney_dis.py

import pandas as pd
import numpy as np


# create header for dataset
header = ['age','bp','sg','al','su','rbc','pc','pcc',
    'ba','bgr','bu','sc','sod','pot','hemo','pcv',
    'wbcc','rbcc','htn','dm','cad','appet','pe','ane',
    'classification']
# read the dataset
df = pd.read_csv("data/chronic_kidney_disease.arff",
        header=None,
        names=header
       )


# print total samples
print("Total samples before cleaning:", len(df))  # 427


# dataset has '?' in it, convert these into NaN
df = df.replace('?', np.nan)
# drop the NaN
df = df.dropna(axis=0, how="any")

# print total samples
print("Total samples after cleaning:", len(df))  # 157
# print 4-rows and 6-columns
print("Partial data\n", df.iloc[0:4, 0:6])
```

Warning: Many times the process of collecting samples could be very expensive (e.g. data received from the satellite which is designed for a one time use), therefore we can not throw the samples out.

In such cases, we need to fill the missing values with some appropriate values e.g. filling with mean values. Also, domain experts can suggest some values based on the other values in the samples.

Listing 1.2: define red/green for chronic/non-chronic disease

```
# kidney_dis.py

import pandas as pd
import numpy as np


# create header for dataset
header = ['age','bp','sg','al','su','rbc','pc','pcc',
    'ba','bgr','bu','sc','sod','pot','hemo','pcv',
    'wbcc','rbcc','htn','dm','cad','appet','pe','ane',
    'classification']
# read the dataset
df = pd.read_csv("data/chronic_kidney_disease.arff",
        header=None,
        names=header
       )


# # print total samples
# print("Total samples before cleaning:", len(df))  # 427


# dataset has '?' in it, convert these into NaN
df = df.replace('?', np.nan)
# drop the NaN
df = df.dropna(axis=0, how="any")

# # print total samples
# print("Total samples after cleaning:", len(df))  # 157
# # print 4-rows and 6-columns
# print("Partial data\n", df.iloc[0:4, 0:6])


# save column 'classification' in variable 'targets'
targets = df['classification'].astype('category')

# # print target
# print_target = ['ckd' if i=='ckd' else 'nckd' for i in targets]
# # ['ckd', 'ckd', 'ckd'] ['nckd', 'nckd']
# print(print_target[0:3], print_target[-3:-1])

# for plotting, assign color to targets
# red: disease,  green: no disease
label_color = ['red' if i=='ckd' else 'green' for i in targets]
# ['red', 'red', 'red'] ['green', 'green']
print(label_color[0:3], label_color[-3:-1])
```

1.3.3 Remove categorical data

Note that, we can use only numeric data in PCA algorithms, therefore we need to remove/modify the categorical data. In the below code, we removed the categorical data.

Warning: Removing the categorical data from the samples is a bad idea as we are wasting the information.

Listing 1.3: Remove categorical data

```
# kidney_dis.py

import pandas as pd
import numpy as np

# create header for dataset
header = ['age','bp','sg','al','su','rbc','pc','pcc',
    'ba','bgr','bu','sc','sod','pot','hemo','pcv',
    'wbcc','rbcc','htn','dm','cad','appet','pe','ane',
    'classification']
# read the dataset
df = pd.read_csv("data/chronic_kidney_disease.arff",
        header=None,
        names=header
       )


# # print total samples
# print("Total samples before cleaning:", len(df))  # 427


# dataset has '?' in it, convert these into NaN
df = df.replace('?', np.nan)
# drop the NaN
df = df.dropna(axis=0, how="any")

# # print total samples
# print("Total samples after cleaning:", len(df))  # 157
# # print 4-rows and 6-columns
# print("Partial data\n", df.iloc[0:4, 0:6])


# save column 'classification' in variable 'targets'
targets = df['classification'].astype('category')

# # print target
# print_target = ['ckd' if i=='ckd' else 'nckd' for i in targets]
# # ['ckd', 'ckd', 'ckd'] ['nckd', 'nckd']
# print(print_target[0:3], print_target[-3:-1])

# for plotting, assign color to targets
# red: disease,  green: no disease
label_color = ['red' if i=='ckd' else 'green' for i in targets]
# ['red', 'red', 'red'] ['green', 'green']
print(label_color[0:3], label_color[-3:-1])


print("Partial data before processing\n", df.iloc[0:4, 0:7]) # print partial data

# list of categorical features
categorical_ = ['rbc', 'pc', 'pcc', 'ba', 'htn',
        'dm', 'cad', 'appet', 'pe', 'ane'
        ]

# drop the "categorical" features
# drop the classification column
```

(continues on next page)

(continued from previous page)

```
df = df.drop(labels=['classification'], axis=1)
# drop using 'inplace' which is equivalent to df = df.drop()
df.drop(labels=categorical_, axis=1, inplace=True)
print("Partial data after processing\n", df.iloc[0:4, 0:7]) # print partial data
```

1.3.4 Apply PCA analysis

Now data is prepared and we are ready to use the PCA algorithm.

Note: It is quite a straight forward process. All we need to do is "use the algorithm" which is already implemented in the library. In this example we are using "sklearn" library, but there are other libraries as well e.g. TensorFlow, Caffe and Theano etc.

Warning:

- Don't stick to one library or argue that a library is better than the other library.
- Try to use the best features of every library.
- In the end, everything is number (multi dimensional array) in data analysis, therefore we can transform the data so that it can be used by other libraries.

Listing 1.4: PCA analysis

```
# kidney_dis.py

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

# create header for dataset
header = ['age','bp','sg','al','su','rbc','pc','pcc',
    'ba','bgr','bu','sc','sod','pot','hemo','pcv',
    'wbcc','rbcc','htn','dm','cad','appet','pe','ane',
    'classification']
# read the dataset
df = pd.read_csv("data/chronic_kidney_disease.arff",
        header=None,
        names=header
       )


# # print total samples
# print("Total samples before cleaning:", len(df))  # 427


# dataset has '?' in it, convert these into NaN
df = df.replace('?', np.nan)
# drop the NaN
df = df.dropna(axis=0, how="any")

# # print total samples
# print("Total samples after cleaning:", len(df))  # 157
# # print 4-rows and 6-columns
# print("Partial data\n", df.iloc[0:4, 0:6])

```

(continues on next page)

(continued from previous page)

```
34
35  # save column 'classification' in variable 'targets'
36  targets = df['classification'].astype('category')
37
38  # # print target
39  # print_target = ['ckd' if i=='ckd' else 'nckd' for i in targets]
40  # # ['ckd', 'ckd', 'ckd'] ['nckd', 'nckd']
41  # print(print_target[0:3], print_target[-3:-1])
42
43  # for plotting, assign color to targets
44  # red: disease,  green: no disease
45  label_color = ['red' if i=='ckd' else 'green' for i in targets]
46  # ['red', 'red', 'red'] ['green', 'green']
47  print(label_color[0:3], label_color[-3:-1])
48
49
50  print("Partial data before processing\n", df.iloc[0:4, 0:7]) # print partial data
51
52  # list of categorical features
53  categorical_ = ['rbc', 'pc', 'pcc', 'ba', 'htn',
54          'dm', 'cad', 'appet', 'pe', 'ane'
55          ]
56
57  # drop the "categorical" features
58  # drop the classification column
59  df = df.drop(labels=['classification'], axis=1)
60  # drop using 'inplace' which is equivalent to df = df.drop()
61  df.drop(labels=categorical_, axis=1, inplace=True)
62  print("Partial data after processing\n", df.iloc[0:4, 0:7]) # print partial data
63
64
65  # PCA analysis
66  pca = PCA(n_components=2)
67  pca.fit(df)
68  T = pca.transform(df) # transformed data
69  # change 'T' to Pandas-DataFrame to plot using Pandas-plots
70  T = pd.DataFrame(T)
71
72  # plot the data
73  T.columns = ['PCA component 1', 'PCA component 2']
74  T.plot.scatter(x='PCA component 1', y='PCA component 2', marker='o',
75          alpha=0.7, # opacity
76          color=label_color,
77          title="red: ckd, green: not-ckd" )
78  plt.show()
```

Note: It is easy to visualize the data in 2D and 3D format (usually 2D); there "pca = PCA(n_components=2)" is used in the code, which will reduce the dataset into 2 components.

1.3.5 PCA limitation

PCA is dominated by 'high variance features'. Therefore features should be normalized before using the PCA model. In the below code 'StandardScalar' preprocessing module is used to normalized the features, which sets the 'mean=0' and 'variance=1' for all the features.

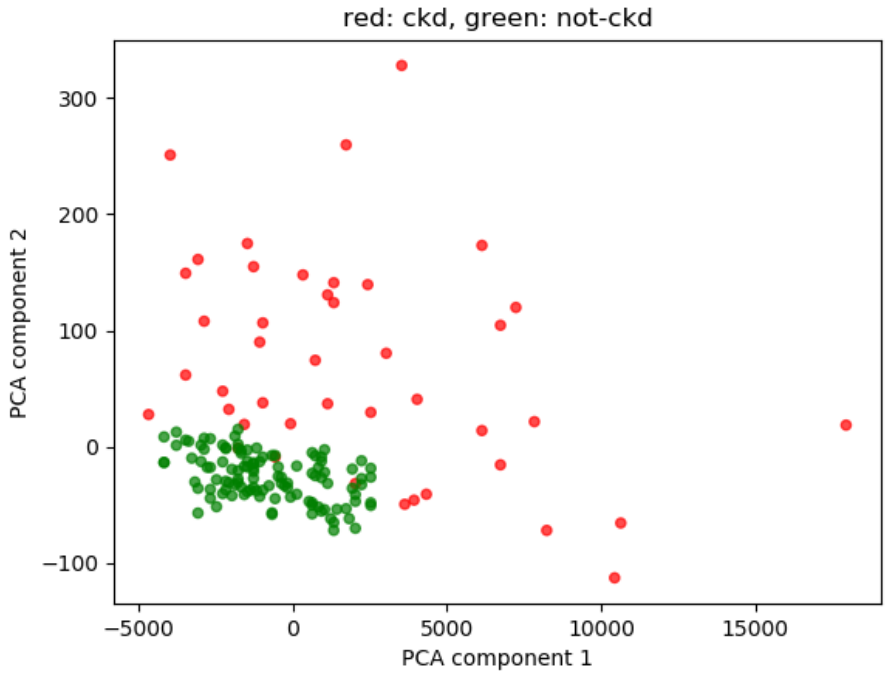

Fig. 1.1: Result of PCA analysis

Listing 1.5: PCA analysis

```
# kidney_dis.py

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn import preprocessing


# create header for dataset
header = ['age','bp','sg','al','su','rbc','pc','pcc',
    'ba','bgr','bu','sc','sod','pot','hemo','pcv',
    'wbcc','rbcc','htn','dm','cad','appet','pe','ane',
    'classification']
# read the dataset
df = pd.read_csv("data/chronic_kidney_disease.arff",
        header=None,
        names=header
       )


# # print total samples
# print("Total samples before cleaning:", len(df))  # 427


# dataset has '?' in it, convert these into NaN
df = df.replace('?', np.nan)
# drop the NaN
df = df.dropna(axis=0, how="any")

```

(continues on next page)

(continued from previous page)

```
# # print total samples
# print("Total samples after cleaning:", len(df))  # 157
# # print 4-rows and 6-columns
# print("Partial data\n", df.iloc[0:4, 0:6])


# save column 'classification' in variable 'targets'
targets = df['classification'].astype('category')

# # print target
# print_target = ['ckd' if i=='ckd' else 'nckd' for i in targets]
# # ['ckd', 'ckd', 'ckd'] ['nckd', 'nckd']
# print(print_target[0:3], print_target[-3:-1])

# for plotting, assign color to targets
# red: disease,  green: no disease
label_color = ['red' if i=='ckd' else 'green' for i in targets]
# ['red', 'red', 'red'] ['green', 'green']
print(label_color[0:3], label_color[-3:-1])


print("Partial data before processing\n", df.iloc[0:4, 0:7]) # print partial data

# list of categorical features
categorical_ = ['rbc', 'pc', 'pcc', 'ba', 'htn',
        'dm', 'cad', 'appet', 'pe', 'ane'
        ]

# drop the "categorical" features
# drop the classification column
df = df.drop(labels=['classification'], axis=1)
# drop using 'inplace' which is equivalent to df = df.drop()
df.drop(labels=categorical_, axis=1, inplace=True)
print("Partial data after processing\n", df.iloc[0:4, 0:7]) # print partial data


# StandardScaler: mean=0, variance=1
df = preprocessing.StandardScaler().fit_transform(df)


# PCA analysis
pca = PCA(n_components=2)
pca.fit(df)
T = pca.transform(df) # transformed data
# change 'T' to Pandas-DataFrame to plot using Pandas-plots
T = pd.DataFrame(T)

# plot the data
T.columns = ['PCA component 1', 'PCA component 2']
T.plot.scatter(x='PCA component 1', y='PCA component 2', marker='o',
        alpha=0.7, # opacity
        color=label_color,
        title="red: ckd, green: not-ckd" )
plt.show()
```

Note: In Fig. 1.2, we can see that the performance is significantly improved just by adding one line.

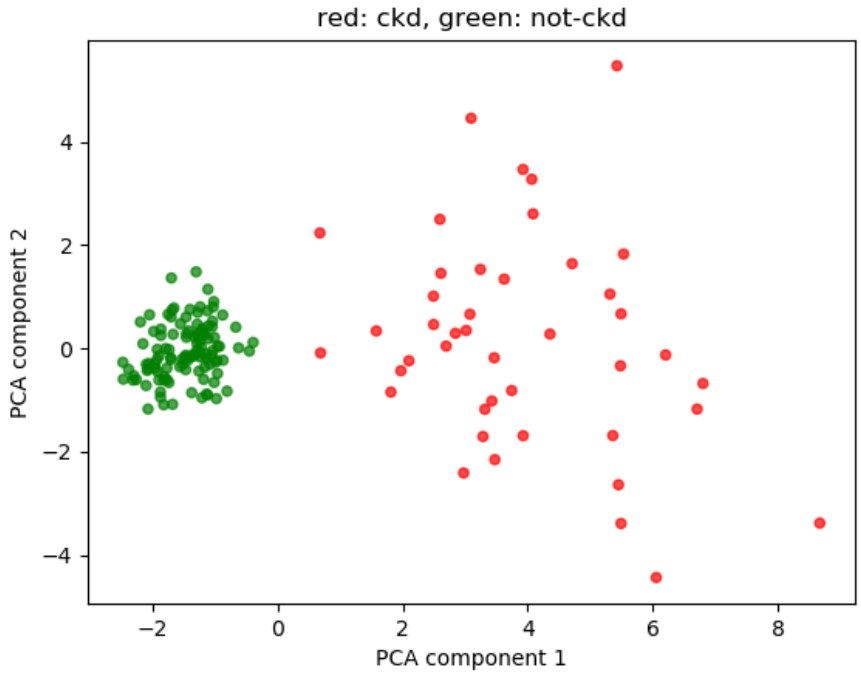

Fig. 1.2: Result of PCA analysis after preprocessing the data

1.3.6 Convert 'categorical' features to 'numeric' features

In Listing 1.3 we removed the categorical features. But the performance can be further improved if we can use these categorical features in the PCA algorithm (as we will have more samples). The 'categorical' features can be represented as numbers but assigning a number can be quite a lengthy process; hence it is better to use some existing library for it. In the below code, we used Pandas library for the same.

Listing 1.6: PCA analysis

```
# kidney_dis.py

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn import preprocessing


# create header for dataset
header = ['age','bp','sg','al','su','rbc','pc','pcc',
    'ba','bgr','bu','sc','sod','pot','hemo','pcv',
    'wbcc','rbcc','htn','dm','cad','appet','pe','ane',
    'classification']
# read the dataset
df = pd.read_csv("data/chronic_kidney_disease.arff",
        header=None,
        names=header
       )


# # print total samples
# print("Total samples before cleaning:", len(df))  # 427

```

(continues on next page)

(continued from previous page)

```

# dataset has '?' in it, convert these into NaN
df = df.replace('?', np.nan)
# drop the NaN
df = df.dropna(axis=0, how="any")

# # print total samples
# print("Total samples after cleaning:", len(df))  # 157
# # print 4-rows and 6-columns
# print("Partial data\n", df.iloc[0:4, 0:6])


# save column 'classification' in variable 'targets'
targets = df['classification'].astype('category')

# # print target
# print_target = ['ckd' if i=='ckd' else 'nckd' for i in targets]
# # ['ckd', 'ckd', 'ckd'] ['nckd', 'nckd']
# print(print_target[0:3], print_target[-3:-1])

# for plotting, assign color to targets
# red: disease,  green: no disease
label_color = ['red' if i=='ckd' else 'green' for i in targets]
# ['red', 'red', 'red'] ['green', 'green']
print(label_color[0:3], label_color[-3:-1])


print("Partial data before processing\n", df.iloc[0:4, 0:7]) # print partial data

# list of categorical features
categorical_ = ['rbc', 'pc', 'pcc', 'ba', 'htn',
        'dm', 'cad', 'appet', 'pe', 'ane'
        ]

# drop the "categorical" features
# drop the classification column
df = df.drop(labels=['classification'], axis=1)
# drop using 'inplace' which is equivalent to df = df.drop()
# df.drop(labels=categorical_, axis=1, inplace=True)
# convert categorical features into dummy variable
df = pd.get_dummies(df, columns=categorical_)
# print("Partial data\n", df.iloc[0:4, 0:6]) # print partial data
print("Partial data after processing\n", df.iloc[0:4, 0:7]) # print partial data


# StandardScaler: mean=0, variance=1
df = preprocessing.StandardScaler().fit_transform(df)


# PCA analysis
pca = PCA(n_components=2)
pca.fit(df)
T = pca.transform(df) # transformed data
# change 'T' to Pandas-DataFrame to plot using Pandas-plots
T = pd.DataFrame(T)

# plot the data
T.columns = ['PCA component 1', 'PCA component 2']
T.plot.scatter(x='PCA component 1', y='PCA component 2', marker='o',
        alpha=0.7, # opacity
        color=label_color,
```

(continues on next page)

(continued from previous page)

```
        title="red: ckd, green: not-ckd" )
plt.show()
```

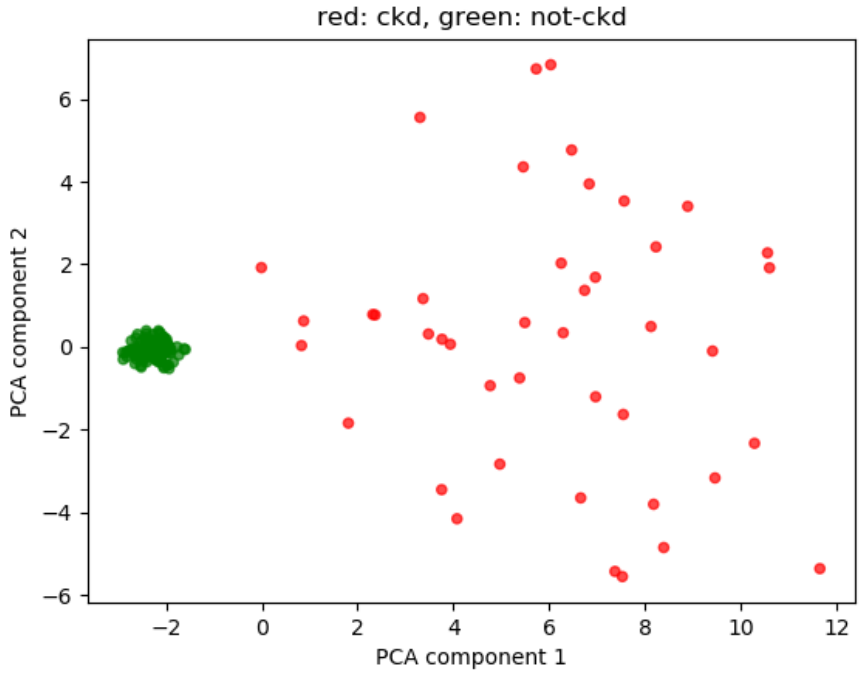

Fig. 1.3: Result of PCA analysis after converting categorical features to numerical feature

1.4 Summary

Following are the steps which are required for data analysis,

- Read and clean the data.
- Reformat the data so that it can be used by the algorithm.
- Process the data to enhance the performance of the algorithm.
- Check the performance of the algorithm. If it is not good then try another algorithm.
- Also, we should try to use all the samples for training. In the current example, we drop several samples which can further improve the training of the algorithm.

Note: We did not cover various topics e.g.,

- What is training and test dataset.
- How to create correct dataset for training and test samples.
- How to select the best features from the dataset.
- Various method to visualize the data i.e. density plots, histograms and Box & Whisker plot.
- How to check the performance of training e.g. Mean square error and Receiver operating characteristic etc.
