02PreProcessing

November 10, 2018

1 前處理-資料清理

1. missing data
   - 平均值、標準差、屆在平均值與標準差之間的亂數
2. range 差異過大的資料
   - 偵測 (identity): PCA
   - 處理: 取 log e.g. 10ˆ1 => 1, 10ˆ6 => 6
3. 資料不一致的問題
   - domain knowledge e.g. 年紀為負的
4. 正規化 (Normalize)
   - L1 Norm(穩定: 水平調整較少):
     - $\|\|_1 = \sum ||$
   - L2 Norm(強健: 較能對抗 outlier):
     - $\|\|_2 = \frac{}{\sqrt{(\sum^2)}}$
5. 類別型資料的處理
   - 自然語言 (NLP): 先轉成類別型資料
   - onehot encoding

0	1	2
[1,0,0]	[0,1,0]	[0,0,1]

7. Feature 產生工具
   - PolynomialFeatures: (_1, _2)=> (1, _1, _2, _1ˆ2, _1 _2, _2ˆ2)
8. 議題:
   - 請問取 log 與 normalize 有什麼差別?

2 IMPORT & DATA

```
In [1]: import pandas as pd
        import numpy as np
        from collections import Counter
        import re
        import numpy as np

        from sklearn import preprocessing

        import matplotlib.pyplot as plt
        from mpl_toolkits.mplot3d import Axes3D
        from sklearn.decomposition import PCA

        import os
        import random
        import math
```

```
In [2]: df = pd.read_csv('train.csv')

        # 請查看 df.info()
        # 並找出共有幾種型別，以及哪一些欄位有 null 值
        #=============your works starts===============#
        df_info =
        #==============your works ends================#
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

```
In [3]: # 請查看 df.describe()
        # 請透過 mean 關注每一個變數的 scale
        #=============your works starts===============#
        df_describe =
```

```
#==============your works ends================#
df_describe
```

Out[3]:

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

In [4]:

```
# 請透過 head() 查看 df 的頭 5 行
#=============your works starts===============#
df_head =
#==============your works ends================#

df_head
```

Out[4]:

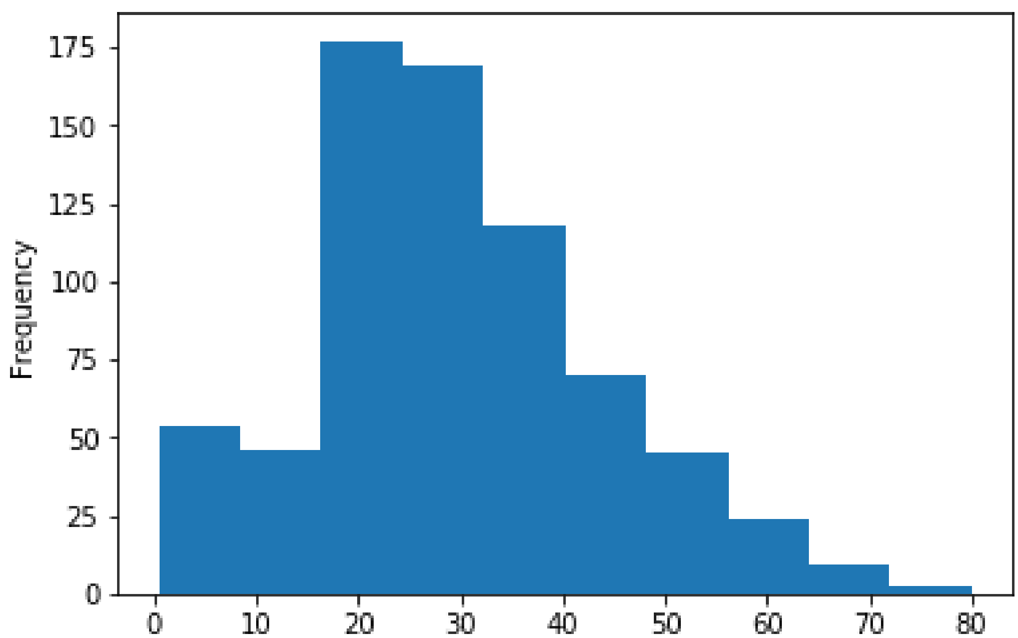
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
```

```
3       0             113803   53.1000   C123         S
4       0             373450    8.0500    NaN         S
```

3 PREPROCESSING

3.1 Age - Fill in missing values

```
In [5]: # 查看 Age 的分布狀況 (hint: df['Age'].plot('hist'))
        #============your works starts===============#
        age_ax =
        #==============your works ends================#

        plt.show()
```




```
In [6]: # 作法一: 取平均值
        #============your works starts===============#
        avg_age =
        #==============your works ends================#


        print("avg_age", avg_age)
        # avg_age 29.69911764705882
```

```
avg_age 29.69911764705882
```

In [7]:
```python
# 作法二：取中位數
#=============your works starts===============#
median_age =
#==============your works ends================#


print("median_age", median_age)
# median_age 28.0
```

```
median_age 28.0
```

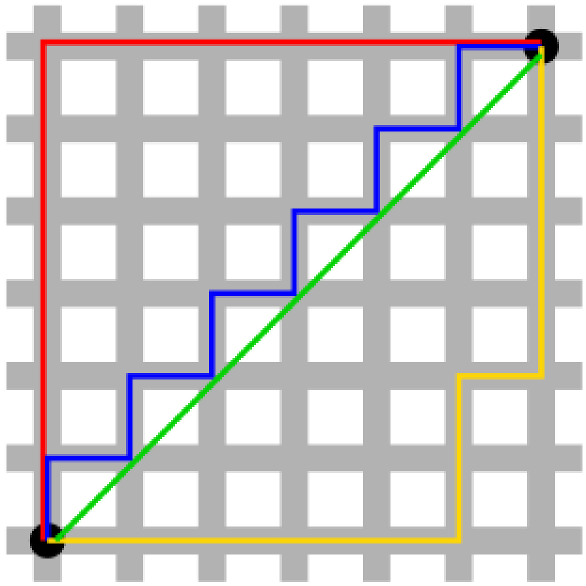
In [8]:
```python
# 作法三：用相同的分布產生亂數塞入 (hint: 使用 np.random.randint)
std = df['Age'].std()
mean = df['Age'].mean()
size = len(df[pd.isnull(df['Age'])])
np.random.seed(1212)
#=============your works starts===============#
random_age =
#==============your works ends================#


print("random_age", random_age)
print("len(random_age)", len(random_age))
# random_age [23 41 37 17 31 20 28 24 16 42 33 19 22 20 29 15 32 16 35 40 35 34 26 27
# len(random_age) 177
```

```
random_age [23 41 37 17 31 20 28 24 16 42 33 19 22 20 29 15 32 16 35 40 35 34 26 27
 37 28 30 23 31 33 42 30 25 21 29 15 21 16 39 39 21 31 31 37 31 30 23 41
 30 35 33 21 31 28 39 37 31 29 29 40 16 43 20 29 36 22 27 41 32 24 35 23
 29 43 33 43 31 34 34 28 27 40 29 35 27 20 40 37 16 29 29 39 20 17 20 35
 24 42 34 33 26 38 42 31 30 40 34 16 35 16 34 24 43 29 22 29 20 43 29 38
 37 39 35 42 40 19 32 17 25 36 15 26 31 23 19 24 34 39 39 19 17 28 16 35
 20 16 29 18 34 43 16 28 30 42 27 25 36 19 22 43 37 38 30 15 32 38 41 21
 26 33 20 19 21 29 40 30 28]
len(random_age) 177
```

In [9]:
```python
df['avg_age'] = df['Age']
df.loc[pd.isnull(df['Age']), 'avg_age'] = avg_age
df['median_age'] = df['Age']
df.loc[pd.isnull(df['Age']), 'median_age'] = median_age
df['random_age'] = df['Age']
df.loc[pd.isnull(df['Age']), 'random_age'] = random_age

df.loc[pd.isnull(df['Age']), ['avg_age', 'median_age', 'random_age']].head()
```

Out[9]:

	avg_age	median_age	random_age
5	29.699118	28.0	23.0



l1_l2_norm

17	29.699118	28.0	41.0
19	29.699118	28.0	37.0
26	29.699118	28.0	17.0
28	29.699118	28.0	31.0

3.2 Age - Normalize

- L1 Normalization: $||x||_2 = \sqrt{(\sum_i x_i^2)} = \sqrt{x_1^2 + x_2^2 + \ldots + x_i^2}$
- L2 Normalization: $||x||_1 = \sum_i |x_i| = |x_1| + |x_2| + \ldots + |x_i|$
- difference

In [10]:
```
# 請寫出 L1 Normaliaze 的 function
def normalize_l1(X):
    """if type(X) == np.array, and X has two dimensiions"""
    #=============your works starts===============#
    l1_x =
    X =
    #==============your works ends================#
    return X

X = [[ 1., -1.,  2.],
     [ 2.,  0.,  0.],
     [ 0.,  1., -1.]]
X_normalized = normalize_l1(X)
print(X_normalized)
# [[ 0.25 -0.25  0.5 ]
```

```
        #  [ 0.5   0.    0.  ]
        #  [ 0.    0.25 -0.25]]
```

```
[[ 0.25 -0.25  0.5 ]
 [ 0.5   0.    0.  ]
 [ 0.    0.25 -0.25]]
```

```
In [11]: # 請寫出 L2 Normaliaze 的 function
         def normalize_l2(X):
             """if type(X) == np.array, and X has two dimensiions"""
             #=============your works starts===============#
             l2_x =
             X =
             #==============your works ends================#
             return X

         X_normalized = normalize_l2(X)
         print(X_normalized)
         # [[ 0.28867513 -0.28867513  0.57735027]
         #  [ 0.57735027  0.          0.        ]
         #  [ 0.          0.28867513 -0.28867513]]
```

```
[[ 0.28867513 -0.28867513  0.57735027]
 [ 0.57735027  0.          0.        ]
 [ 0.          0.28867513 -0.28867513]]
```

```
In [12]: X = [[ 1., -1.,  2.],
              [ 2.,  0.,  0.],
              [ 0.,  1., -1.]]

         # 請使用 preprocessing.normalize(X, norm='l1') 比較，與我們自己寫的 normalize function
         #=============your works starts===============#
         X_normalized =
         #==============your works ends================#

         print(X_normalized)
         # sklearn l1_norm
         # [[ 0.25 -0.25  0.5 ]
         #  [ 1.    0.    0.  ]
         #  [ 0.    0.5  -0.5 ]]
```

```
[[ 0.25 -0.25  0.5 ]
 [ 1.    0.    0.  ]
 [ 0.    0.5  -0.5 ]]
```

```
In [13]: avg_age_l1 = normalize_l1(df['avg_age'].values)
         avg_age_l2 = normalize_l2(df['avg_age'].values)
```

```
        df['avg_age_l1'] = avg_age_l1
        df['avg_age_l2'] = avg_age_l2

        df[['avg_age', 'avg_age_l1', 'avg_age_l2']].head()
```

Out[13]:

	avg_age	avg_age_l1	avg_age_l2
0	22.0	0.000831	0.022735
1	38.0	0.001436	0.039270
2	26.0	0.000983	0.026869
3	35.0	0.001323	0.036170
4	35.0	0.001323	0.036170

3.3 Cabin - NLP category

```
In [14]: # 整理出每一個 Cabin 的個數並排序 (hint:Counter(), sorted())
         #=============your works starts===============#
         sorted_cabin_counter =
         #==============your works ends================#


         print(sorted_cabin_counter[:10])
         # [('A10', 1), ('A14', 1), ('A16', 1), ('A19', 1), ('A20', 1), ('A23', 1), ('A24', 1)

[('A10', 1), ('A14', 1), ('A16', 1), ('A19', 1), ('A20', 1), ('A23', 1), ('A24', 1), ('A26', 1
```

```
In [15]: # 抓出第一個 char 出來分類，並轉成 int 類別 (hint:Counter(), sorted())
         #=============your works starts===============#
         new_Cabin =
         mapping_dict =
         new_Cabin_int =
         #==============your works ends================#


         print(new_Cabin.values[:10])
         print(mapping_dict)
         print(new_Cabin_int[:10])
         # ['n' 'C' 'n' 'C' 'n' 'n' 'E' 'n' 'n' 'n']
         # {'n': 0, 'B': 1, 'T': 2, 'E': 3, 'C': 4, 'F': 5, 'G': 6, 'D': 7, 'A': 8}
         # [0, 4, 0, 4, 0, 0, 3, 0, 0, 0]

['n' 'C' 'n' 'C' 'n' 'n' 'E' 'n' 'n' 'n']
{'T': 0, 'A': 1, 'F': 2, 'G': 3, 'n': 4, 'C': 5, 'E': 6, 'D': 7, 'B': 8}
[4, 5, 4, 5, 4, 4, 6, 4, 4, 4]
```

```
In [16]: df['cabin_cat'] = new_Cabin_int
         df[['Cabin', 'cabin_cat']].head()
```

```
Out[16]:   Cabin  cabin_cat
         0   NaN          4
         1   C85          5
         2   NaN          4
         3  C123          5
         4   NaN          4
```

3.4 Sex - Category

```
In [17]: # 請算出 Sex 共有幾個類別，每一個類別共出現幾次 (hint:Counter)
         #=============your works starts===============#
         counter =
         #==============your works ends================#


         print(counter)
         #Counter({'male': 577, 'female': 314})
```

```
Counter({'male': 577, 'female': 314})
```

```
In [18]: # 創造出一個與 df['Sex'] 等長的 array，並將 df['Sex'] 中的 male 換成 1，female 換成 0
         #=============your works starts===============#
         sex_mapping =
         sex_cat =
         #==============your works ends================#


         print("Counter(sex_cat)", Counter(sex_cat))
         #Counter(sex_cat) Counter({1: 577, 0: 314})
```

```
Counter(sex_cat) Counter({1: 577, 0: 314})
```

```
In [19]: df['sex_cat'] = sex_cat
         Counter(df['sex_cat'])

Out[19]: Counter({1: 577, 0: 314})
```

3.5 Ticket - Category

```
In [20]: # 整理出每一個 Ticket 的個數並排序 (hint:Counter(), sorted())
         #=============your works starts===============#
         sorted_ticket_counter =
         #==============your works ends================#

         print(sorted_ticket_counter)
         # [('110152', 3), ('110413', 3), ('110465', 2), ('110564', 1), ('110813', 1), ('11124
```

```
[('110152', 3), ('110413', 3), ('110465', 2), ('110564', 1), ('110813', 1), ('111240', 1), ('11
```

```
In [21]: # ticket
         ticket_cat = {}
         for ticket in df['Ticket']:
             if ticket.isdigit():
                 ticket_cat[ticket] = 1
             elif ticket.startswith('A'):
                 ticket_cat[ticket] = 2
             elif ticket.startswith('C'):
                 ticket_cat[ticket] = 3
             elif ticket.startswith('F'):
                 ticket_cat[ticket] = 4
             elif ticket.startswith('P'):
                 ticket_cat[ticket] = 5
             elif ticket.startswith('SOTON'):
                 ticket_cat[ticket] = 6
             elif ticket.startswith('STON'):
                 ticket_cat[ticket] = 7
             elif ticket.startswith('S'):
                 ticket_cat[ticket] = 8
             elif ticket.startswith('W'):
                 ticket_cat[ticket] = 9
             else:
                 ticket_cat[ticket] = 0
         df['ticket_cat'] = df['Ticket'].apply(ticket_cat.get)
         print(Counter(df['ticket_cat']))
```

```
Counter({1: 661, 5: 65, 3: 47, 8: 30, 2: 29, 7: 18, 6: 17, 9: 13, 4: 7, 0: 4})
```

3.6 Embarked - Category

```
In [22]: # 整理出每一個 Embarked 的個數並排序 (hint:Counter(), sorted())
         #=============your works starts===============#
         sorted_embarked_counter =
         #==============your works ends================#

         print(sorted_embarked_counter)
         # [('C', 168), ('Q', 77), ('S', 644), ('nan', 2)]
```

```
[('C', 168), ('Q', 77), ('S', 644), ('nan', 2)]
```

```
In [23]: # 創造 embarked 的類別對應 dict
         #=============your works starts===============#
         embarked_cat =
         #==============your works ends================#
```

```
    print(embarked_cat)
    #{nan: 0, 'S': 1, 'Q': 2, 'C': 3}
```

```
{nan: 0, 'C': 1, 'S': 2, 'Q': 3}
```

```
In [24]: # 轉換 embarked 為數字類別
         #=============your works starts===============#
         df['embarked_cat'] =
         #==============your works ends================#

         print(Counter(df['embarked_cat']))
         #Counter({1: 644, 3: 168, 2: 77, 0: 2})
```

```
Counter({2: 644, 1: 168, 3: 77, 0: 2})
```

3.7 Title - NLP category

```
In [25]: # 請找到位在"," 以及"." 的所有字並將" ", ".", "," 去掉 (hint: re.findall(), str.replace
         def find_call(name):
             #=============your works starts===============#
             name =
             #==============your works ends================#
             return name

         title_cat_series = df['Name'].apply(find_call)
         print(title_cat_series.values[:10])
         #['Mr' 'Mrs' 'Miss' 'Mrs' 'Mr' 'Mr' 'Mr' 'Master' 'Mrs' 'Mrs']
```

```
['Mr' 'Mrs' 'Miss' 'Mrs' 'Mr' 'Mr' 'Mr' 'Master' 'Mrs' 'Mrs']
```

```
In [26]: title_mapping= {
             'Ms':"Miss",
             'Mlle':"Miss",
             'Miss':"Miss",
             'Mrs':"Mrs",
             'Mme':"Mrs",
             'MrsMartin(ElizabethL':"Mrs",
             'Mr':"Mr"
             }

         title_cat = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}

         def process_title(call):
             if title_cat.get(call):
```

```
            return title_cat.get(call)
        else:
            return title_cat.get(title_mapping.get(call, "Rare"))

    df['title_cat'] = title_cat_series.apply(process_title)
    print(Counter(df['title_cat']))
```

```
Counter({1: 517, 2: 185, 3: 126, 4: 40, 5: 23})
```

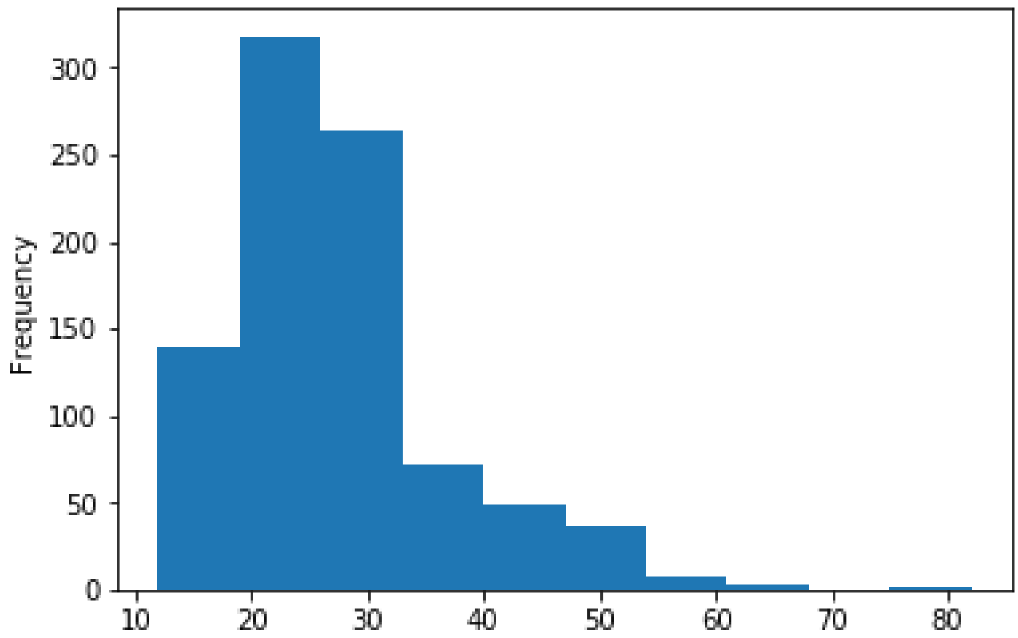
3.8 Title - Length

```
In [27]: # 算出 df['Name'] 中每一個名字的長度並放進一個 array
         #=============your works starts===============#
         name_length =
         #==============your works ends================#


         print(Counter(pd.cut(name_length, bins=10, labels=range(10))))
         # Counter({1: 303, 2: 237, 0: 204, 3: 57, 4: 53, 5: 26, 6: 8, 7: 2, 9: 1})
```

```
Counter({1: 303, 2: 237, 0: 204, 3: 57, 4: 53, 5: 26, 6: 8, 7: 2, 9: 1})
```

```
In [28]: df['name_length'] = name_length
```

```
In [29]: # 劃出每一長度區間次數的長條分布圖 (如長度界在 10~20 之間的有出現約 150 次)(hint: df[col
         #=============your works starts===============#
         df['name_length'].plot('hist')
         #==============your works ends================#


         plt.show()
```

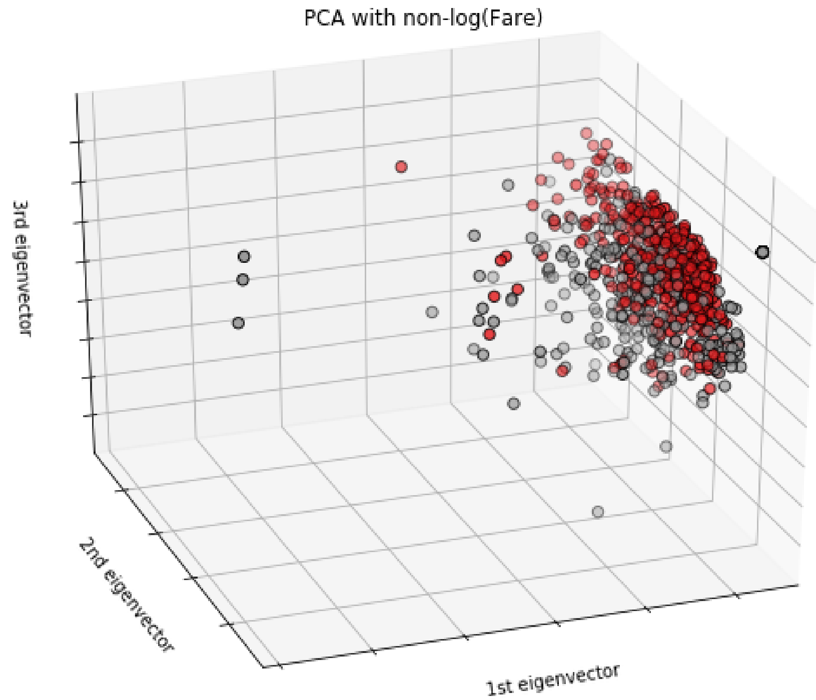


3.9 Fare - PCA, smooth noisy data, feature generation

```
In [30]: # 請找出 dtype 是 np.int64 或 np.float64 且名稱不以'_cat' 結尾的欄位。
         #=============your works starts===============#
         number_cols =
         #==============your works ends================#


         print(number_cols)
         # ['PassengerId', 'Survived', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'avg_age', '

['PassengerId', 'Survived', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'avg_age', 'median_age'
```

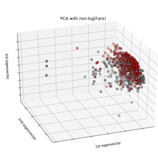
請參照這個連結，劃出以下這張 PCA 圖

```
In [31]: X = df[['Pclass', 'Parch', 'SibSp', 'Fare', 'avg_age', 'name_length']].values
         Y = np.array(df['Survived'])

         #!=============your works starts===============!#
```



PCA_chart

```
#!==============your works ends================!#

plt.show()
```



```
In [32]: # 請找出標準差最大的欄位 ['Pclass', 'Parch', 'SibSp', 'Fare', 'avg_age', 'name_length'
         #!============your works starts==============!#

         #!==============your works ends================!#
```

```
Out[32]:            Pclass       Parch       SibSp        Fare     avg_age  name_length
         count  891.000000  891.000000  891.000000  891.000000  891.000000   891.000000
         mean     2.308642    0.381594    0.523008   32.204208   29.699118    26.965208
         std      0.836071    0.806057    1.102743   49.693429   13.002015     9.281607
         min      1.000000    0.000000    0.000000    0.000000    0.420000    12.000000
         25%      2.000000    0.000000    0.000000    7.910400   22.000000    20.000000
         50%      3.000000    0.000000    0.000000   14.454200   29.699118    25.000000
         75%      3.000000    0.000000    1.000000   31.000000   35.000000    30.000000
         max      3.000000    6.000000    8.000000  512.329200   80.000000    82.000000
```

```
In [33]: # 請找出 Fare 的平均值 (mean)，並填入 df['Fare'] 中
         #=============your works starts===============#
         avg_fare =
         df[pd.isnull(df['Fare'])] =
         #==============your works ends================#
         # df['Fare'].fillna(avg_fare)


         print("avg_fare", avg_fare)
         # 32.204207968574636
         print("number of null of Fare:", len(df[pd.isnull(df['Fare'])]))
         # number of null of Fare: 0

avg_fare 32.204207968574636
number of null of Fare: 0
```

```
In [34]: df['Fare'].describe()

Out[34]: count    891.000000
         mean      32.204208
         std       49.693429
         min        0.000000
         25%        7.910400
         50%       14.454200
         75%       31.000000
         max      512.329200
         Name: Fare, dtype: float64
```

```
In [35]: # 找出 Fare==0 的 row，補上 Fare=1
         #=============your works starts===============#
         df[df['Fare']=
         #==============your works ends================#

         print("number of Fare equals zero:", len(df[df['Fare']==0]))
         # number of Fare equals zero: 0

number of Fare equals zero: 0
```

In [36]:
```
# 請算出 Fare 以 10 為底的 log 值
#=============your works starts===============#
fare_log10 =
#==============your works ends================#


print(fare_log10[:5])
# [0.86033801 1.8529878  0.89899927 1.72509452 0.90579588]
```

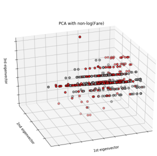
```
[0.86033801 1.8529878  0.89899927 1.72509452 0.90579588]
```

In [37]:
```
df['fare_log10'] = fare_log10
df[['Fare', 'fare_log10']].head()
```

Out[37]:

	Fare	fare_log10
0	7.2500	0.860338
1	71.2833	1.852988
2	7.9250	0.898999
3	53.1000	1.725095
4	8.0500	0.905796

In [38]:
```
X = np.matrix(df[['Parch', 'SibSp', 'avg_age', 'fare_log10']])
Y = np.array(df['Survived'])

fig = plt.figure(1, figsize=(8, 6))
ax = Axes3D(fig, elev=-150, azim=110)

X_reduced = PCA(n_components=3).fit_transform(X)
ax.scatter(X_reduced[:, 0], X_reduced[:, 1], X_reduced[:, 2], c=Y,
           cmap=plt.cm.Set1, edgecolor='k', s=40)
ax.set_title("PCA with non-log(Fare)")
ax.set_xlabel("1st eigenvector")
ax.w_xaxis.set_ticklabels([])
ax.set_ylabel("2nd eigenvector")
ax.w_yaxis.set_ticklabels([])
ax.set_zlabel("3rd eigenvector")
ax.w_zaxis.set_ticklabels([])

plt.show()
```



4 類別型變數 onehot encode

In [39]:
```python
enc = preprocessing.OneHotEncoder()
# 請使用 enc.fit_transform 兩個步驟，onehot encode title_cat
#=============your works starts===============#
title_cat_onehot =
#==============your works ends================#


print(title_cat_onehot[:3])
# [[1. 0. 0. 0. 0.]
#  [0. 0. 1. 0. 0.]
#  [0. 1. 0. 0. 0.]]
```

```
[[1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 1. 0. 0. 0.]]


If you want the future behaviour and silence this warning, you can specify "categories='auto'"
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integer
  warnings.warn(msg, FutureWarning)
```

In [40]:
```python
enc = preprocessing.OneHotEncoder()
# 請使用 enc.fit_transform 兩個步驟，onehot encode embarked_cat
#=============your works starts===============#
embarked_cat_onehot =
#==============your works ends================#

embarked_cat_onehot[:3]
# array([[0., 0., 1., 0.],
#        [0., 1., 0., 0.],
#        [0., 0., 1., 0.]])
```

```
If you want the future behaviour and silence this warning, you can specify "categories='auto'"
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integer
  warnings.warn(msg, FutureWarning)
```

Out[40]:
```
array([[0., 0., 1., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.]])
```

5 PolynomialFeatures

In [41]:
```python
poly = preprocessing.PolynomialFeatures(degree=2)
# 請利用 poly.fit_transform 製造出 fare_log10 的 0 次項、1 次項、2 次項，並把 0 次項拿掉
```

```
#=============your works starts===============#
fare_log10_poly =
#==============your works ends=================#

print(fare_log10_poly[:2])
# [[1.         0.86033801 0.74018149]
#  [1.         1.8529878  3.43356378]]
```

```
[[0.86033801 0.74018149]
 [1.8529878  3.43356378]]
```

In [42]:
```
# 請利用 poly.fit_transform 製造出'fare_log10', 'random_age' 的二項次及其一次交成項
#=============your works starts===============#
age_fare_ploy =
#==============your works ends=================#

print(age_fare_ploy[:2])
# [[8.60338007e-01 8.31383556e-04 7.40181486e-01 7.15270871e-04 6.91198616e-07]
#  [1.85298780e+00 1.43602614e-03 3.43356378e+00 2.66093892e-03 2.06217108e-06]]
```

```
[[8.60338007e-01 8.31383556e-04 7.40181486e-01 7.15270871e-04
  6.91198616e-07]
 [1.85298780e+00 1.43602614e-03 3.43356378e+00 2.66093892e-03
  2.06217108e-06]]
```

5.1 Preprocessing Conclude

In [43]:
```
df.columns
```

Out[43]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked', 'avg_age', 'median_age',
       'random_age', 'avg_age_l1', 'avg_age_l2', 'cabin_cat', 'sex_cat',
       'ticket_cat', 'embarked_cat', 'title_cat', 'name_length', 'fare_log10'],
      dtype='object')
```

In [44]:
```
X = df[['SibSp', 'Parch', 'avg_age_l2', 'sex_cat', 'name_length', 'fare_log10']].value
X = np.concatenate([X, title_cat_onehot, embarked_cat_onehot, age_fare_ploy], axis=1)
Y = df[['Survived']].values

print(X.shape)
print(Y.shape)
```

```
(891, 20)
(891, 1)
```

In [45]:
```
from sklearn import linear_model
from  sklearn.model_selection import train_test_split


X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=1212)

reg = linear_model.LinearRegression()
reg.fit(X_train, Y_train)
predict_prob = reg.predict(X_test)

Y_predict = predict_prob > 0.5
Y_test = Y_test == 1
acc = np.sum(Y_predict == Y_test)/ len(Y_test)
print("Accuracy:", acc)
```

```
Accuracy: 0.8026905829596412
```