机器学习实验报告

实验名称：音乐流行度预测

学生姓名：蒋雨初

学生学号：58121102

完成日期：2023/4/4

任务描述

人类与歌曲和音乐有着紧密的联系。音乐可以改善情绪，减轻疼痛和焦虑，为情感表达提供机会。研究表明，音乐可以在许多方面有益于我们的身体和心理健康。最近，多项研究已经开展，以了解歌曲和其受欢迎程度之间的关系。本数据集中，歌曲用多种指标度量对进行描述并记录为表格，任务是预测歌曲流行度。数据集包含 18835 个样本，每个样本包含 13 个属性与 1 个实值标记，属性包括连续数据属性（10 个）和离散属性（3 个）。要注意数据之间存在多重共线性，这增加了任务的挑战难度。

目标：根据歌曲指标预测歌曲流行度。

实验设置

模型评估

模型的评估方法有留出法、交叉验证法、自助法、留一验证法等方法。在本实验中使用的时最简单的留出法。

留出法直接将数据集 D 划分为两个互斥的集合，其中一个集合用作训练集 S，另一个作为测试集 T。在 S 上训练出模型后，用 T 来评估其测试误差，作为对泛化误差的估计。常见的做法是以训练集: 测试集 $= 2:1 \sim 4:1$的比例进行分割。

性能度量

均方根误差(Mean Square Error，MSE)

均方根误差是最常用的指标，其值越小，模型就越好。公式如下：

$$MSE = \frac{1}{m}\sum_{i=1}^{m}\left(y_{real}^{(i)} - y_{pred}^{(i)}\right)^2$$

但是 MSE 的量纲与原始数据并不同，所以通常使用$RMSE = \sqrt{MSE}$，来评估回归模型的性能。

决定系数R^2

决定系数是衡量观察值与拟合回归线之间的接近程度的指标。因变量会包含一定数量的变化，而 R^2被用来衡量模型解释这种变化的多少。公式如下：

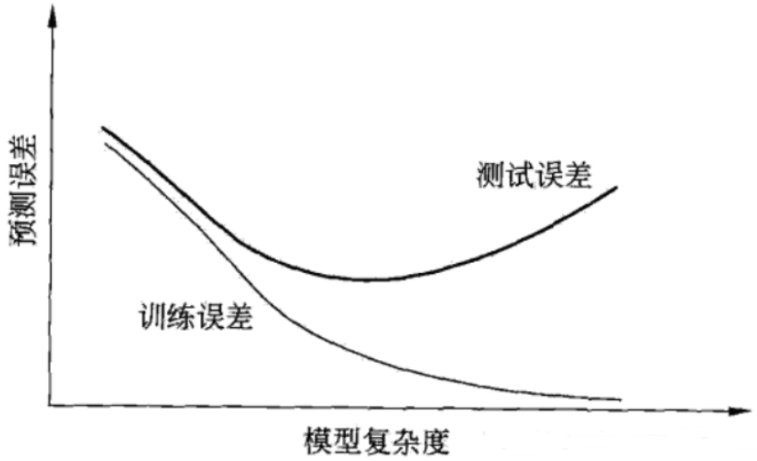
$$R^2 = 1 - \frac{\sum_i\left(y_{real}^{(i)} - y_{pred}^{(i)}\right)^2}{\sum_i\left(\overline{y^{(i)}} - y_{real}^{(i)}\right)^2}$$

根据R^2的取值，来判断模型的好坏，其取值范围为[0,1]：如果结果是 0，说明模型拟合效果很差；如果结果是 1，说明模型能够解释因变量中的所有方差，无错误。一般来说，R^2 越

大，表示模型拟合效果越好。R^2反映的是大概有多准，因为，随着样本数量的增加，R^2必然增加，无法真正定量说明准确程度，只能大概定量[1]。

其他

在模型的选择方面，一个好的模型，除了训练误差和测试误差都应该尽可能小，另一方面它们也应该尽可能接近。



实验过程

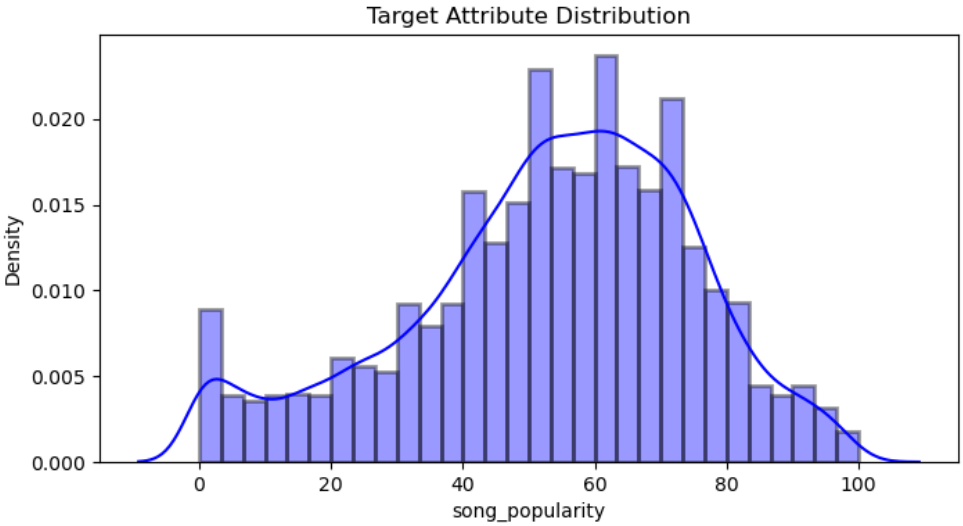
探索性数据分析

在这一步当中，我们需要先对数据集有一个整体的认识，以便进行数据预处理。具体来说，我们想要知道目标属性、每一个连续数据属性的分布和每一个离散属性的取值的分布。此外，还要检查是否有数据缺失。

目标属性

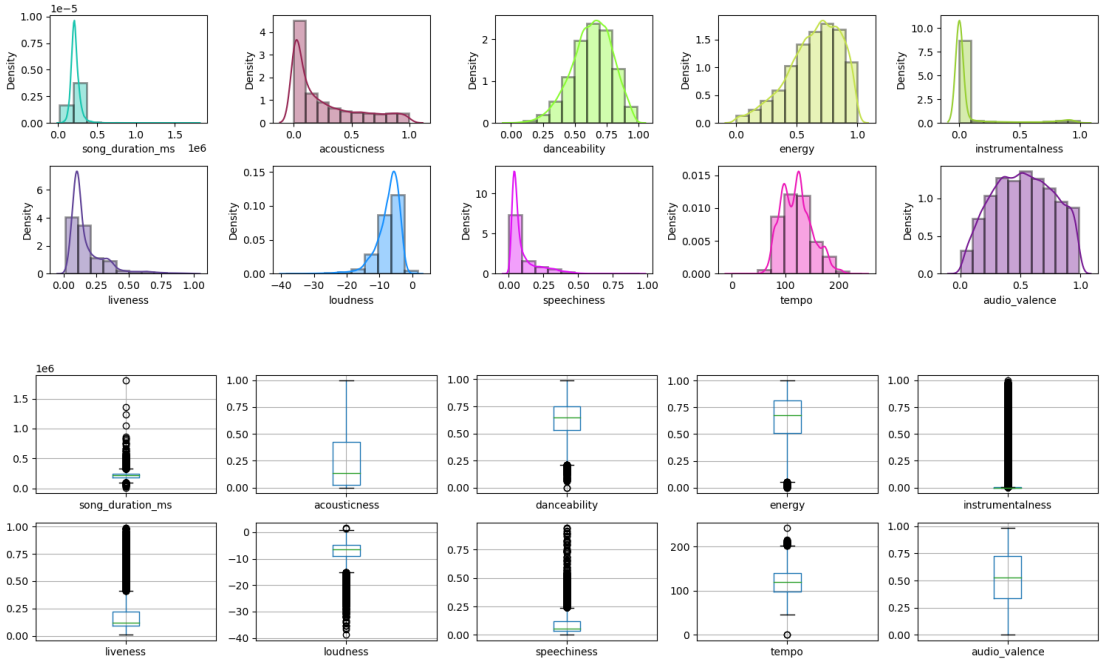
观察目标属性数据分布，可以发现歌曲流行度大致符合正态分布，均值大约是 60。此外，有一些不合理的凸起，这有可能是重复项导致的。

[1] https://blog.csdn.net/u012735708/article/details/84337262

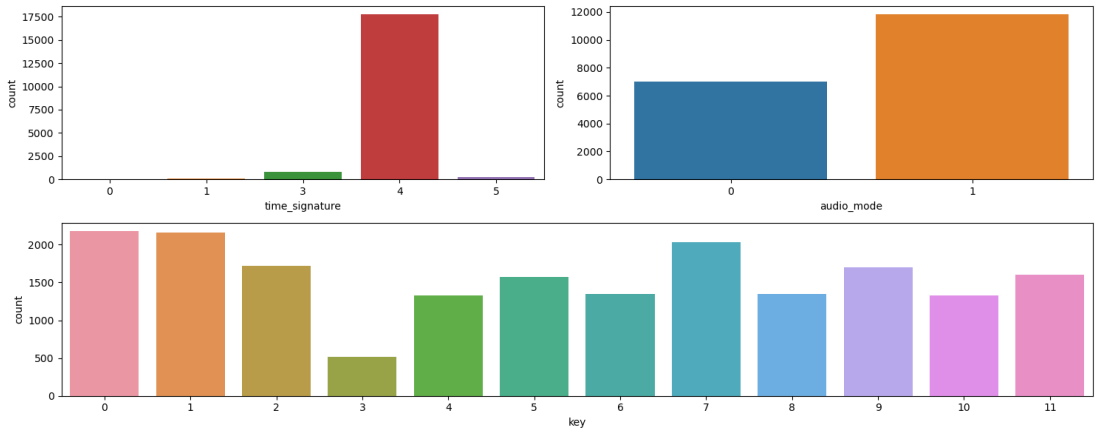


连续数据属性

从箱形图中可以看出有某些属性存在离群值（以圆点标出），这需要在预处理时去除。



离散属性



值缺失检测

在回归问题中，值缺失会严重影响预测结果，所以要提前处理。经检测，没有缺失值。

数据预处理

去重

很容易检测出原始数据中存在许多重复数据，而重复数据会影响样本的分布。有些算法对重复值并不敏感，例如朴素贝叶斯、SVM。而有些算法，例如神经网络、决策树、KNN 就会被影响。特别地，对于本实验中的线性回归，样本重复也是不应发生的，因为模型假定了样本之间是不一样的，也就是计算参数中有个自由度概念，通常 n 减 1。样本重复就减小了它，使得更容易显著。因此我们需要去重。
重复数据共去除 3911 条。

离散数据连续化

这一步主要使用 dummy encoding 或 one hot encoding。audio_mode 属性只由 0 和 1 勾成，故不用处理。而 key 和 time_signature 需要使用 dummy encoding。
在这一步之后，数据变为 27 行（注意，song_name 在导入数据时就已剔除）

离群值去除

这一步根据 IQR(Interquartile Range, 四分位距)[2]完成。在盒形图中已了解到存在离群值，那么我们只保留落于$[Q_1 - 1.5 \times IQR, Q_3 + 1.5 \times IQR]$的数据。

[2] Interquartile Range. [2009-09-18].

在这一步之后数据还有 8930 条。

数据归一化

利用来自 sklearn.preprocessor 的 MinMaxScaler，将连续形数据归一化到[0,1]。公式[3]如下：

$$X_{std} = \frac{X - X.\min(axis = 0)}{X.\max(axis = 0) - X.\min(axis = 0)}$$

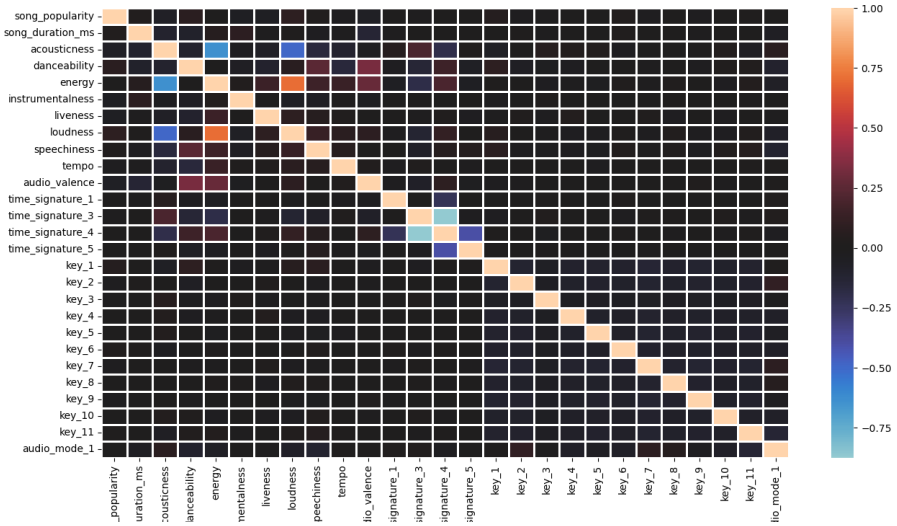
$$X_{scaled} = X_{std} \times (\max - \min) + \min$$

部分数据如下图

	song_popularity	song_duration_ms	acousticness	danceability	energy	instrumentalness
0	73	0.697672	0.005601	0.371762	0.670618	0.004153
1	66	0.510571	0.010454	0.431347	0.849995	0.000000
2	74	0.510571	0.026799	0.313472	0.972726	0.501412
3	56	0.538978	0.000965	0.308290	0.758733	0.000000
4	80	0.588708	0.009083	0.138601	0.946502	0.000261

共线性检测与处理

先用 heatmap 观察相似度系数，可以发现部分数据间存在共线性问题。



共线性（Collinearity）指的就是在拟合线性回归模型的过程中两个或两个以上的 predictors 互相之间存在较强的相关关系。当自变量之间存在共线性时，模型的参数会变得极其不稳定，模型得预测能力会下降。 很难确切区分每个自变量对因变量得影响，因此增加了对于模型结果得解释成本。

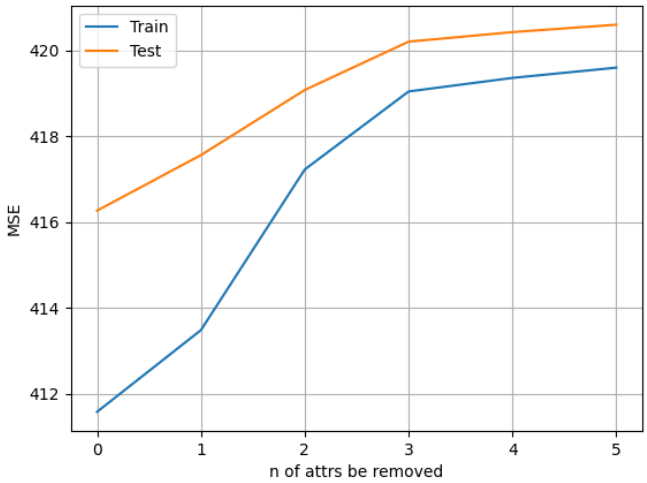
当然，并不是所有的共线性都可以通过观察相似系数得到，还可以考虑计算方差膨胀因子（Variance Inflation Factor，VIF）。VIF 是指解释变量之间存在多重共线性时的方差与不存在多重共线性时的方差之比。VIF 越大，显示共线性越严重。公式[4]如下：

[3] https://blog.csdn.net/GentleCP/article/details/109333753
[4] https://www.zhihu.com/question/270451437/answer/405814593

$$VIF_i = \frac{1}{1 - R_i^2}$$

其中， R_i为第 i 个变量X_i与其他全部变量$X_j(i = 1,2, \dots, k; i \neq j)$的复相关系数，所谓复相关系数即可决系数$R^2$的算术平方根，也即拟合优度的算术平方根。不过这个可决系数R_i^2是指用X_i做因变量，对其他全部X_j做一个新的回归以后得到的可决系数。当$VIF > 5$时我们认为有明显的共线性问题。

为了解决这种问题，我们可以通过手动剔除不合适的属性，根据实验结果，被剔除的属性分别是'time_signature_4', 'energy', 'danceability', 'loudness', 'tempo', 'song_duration_ms'。在这种条件下，得到的训练结果如图。



当然，我们也可以用主成分分析（Principal Component Analysis, PCA）来解决。PCA 是一种使用最广泛的数据降维算法。PCA 的主要思想是将 n 维特征映射到 k 维上，这 k 维是全新的正交特征也被称为主成分，是在原有 n 维特征的基础上重新构造出来的 k 维特征。PCA 的工作就是从原始的空间中顺序地找一组相互正交的坐标轴，新的坐标轴的选择与数据本身是密切相关的。其中，第一个新坐标轴选择是原始数据中方差最大的方向，第二个新坐标轴选取是与第一个坐标轴正交的平面中使得方差最大的，第三个轴是与第 1,2 个轴正交的平面中方差最大的。依次类推，可以得到 n 个这样的坐标轴。通过这种方式获得的新的坐标轴，我们发现，大部分方差都包含在前面 k 个坐标轴中，后面的坐标轴所含的方差几乎为 0。于是，我们可以忽略余下的坐标轴，只保留前面 k 个含有绝大部分方差的坐标轴。事实上，这相当于只保留包含绝大部分方差的维度特征，而忽略包含方差几乎为 0 的特征维度，实现对数据特征的降维处理[5]。

在 PCA 中，保留主成分的原则通常有两种方式：

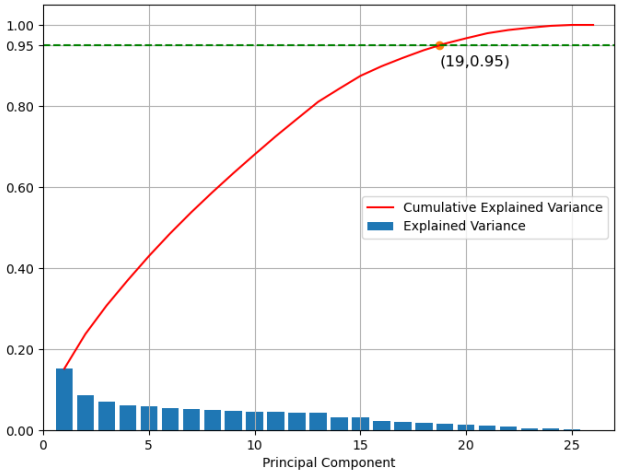
1. 保留方差解释率（explained variance ratio）大于某个阈值的主成分。方差解释率是每个主成分解释的数据方差占总方差的比例。因此，保留方差解释率大于某个阈值的主成分可以尽可能多地保留数据的信息，同时减少特征的数量。

2. 保留主成分的累计方差解释率（cumulative explained variance ratio）大于某个阈值。累计方差解释率是前 k 个主成分的方差解释率之和。这种方法可以在保留足够多的数据信息的同时，减少主成分的数量。

在选择保留主成分的方法时，需要根据具体问题和数据集的特点来决定。一般来说，可以通过绘制方差解释率或累计方差解释率的图像来帮助确定保留的主成分数量。由图中可以

[5] https://zhuanlan.zhihu.com/p/37777074

看出，当设定阈值为 0.95 时，可以考虑保留$k = 19$个主成分。



训练与预测

使用 80%的数据集进行训练，20%的数据集进行测试。

实验结果

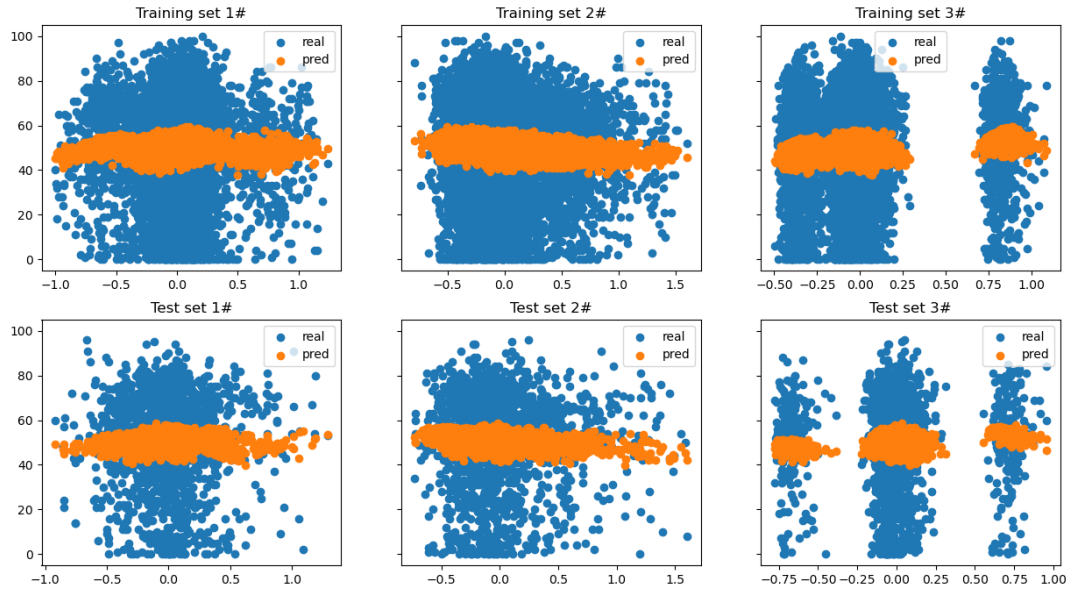
所得到模型的系数与截距：

```
coeffecients [ 0.73176258 -3.1390288  -1.68976287  3.63350686 -0.50598238  0.60791556
  2.26683609 -0.91905383  0.70288648  2.15286472  1.83899824  1.88991101
 -4.43095821 -1.21467123 -6.05738371 -7.07082351 -0.56766173 -1.26462167]
intercepts 50.297486033519554
```

在训练集和测试集上的 MSE 和 R2 分数：

```
MSE of Training set 417.84844102696957
R2-Score of Testing set 0.0225276808346192
MSE of Test set 428.0371320116707
R2-Score of Testing set -0.024763473532357638
```

在训练集和测试集上真实值和预测值的情况：



结果分析

- 原始数据集存在较多的离群值和重复值，这些都是需要删去的
- 多个属性之间存在共线性问题，通过 VIF 可以检测出来，再用 PCA 进行数据降维。
- 使用线性模型 MSE 较大，R2 分数较低，可能并不是很适合这个任务。而训练集与测试集之间 MSE 差距较小，说明这个模型复杂度是合适的。

代码附录

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
from statsmodels.stats.outliers_influence import
variance_inflation_factor

# elegant prepocess
df = pd.read_csv('song_data.csv').drop('song_name',
axis=1).drop_duplicates()
attrs = [cols for cols in df.columns if cols not in
['song_popularity']]
category_attrs = ['time_signature', 'key', 'audio_mode']
numeric_attrs = [
    'danceability', 'energy', 'speechiness', 'audio_valence',
'liveness',
    'acousticness', 'instrumentalness', 'loudness', 'song_duration_ms',
'tempo'
]
df = pd.get_dummies(df, columns=category_attrs[:2])

for attr in numeric_attrs:
    Q1, Q3 = df[attr].quantile(0.25), df[attr].quantile(0.75)
    IQR = Q3 - Q1
    df = df[(Q1 - 1.5 * IQR <= df[attr])
            & (df[attr] <= (Q3 + 1.5 * IQR))].reset_index(drop=True)

scaler = MinMaxScaler()
df[numeric_attrs] = scaler.fit_transform(df[numeric_attrs])
```

```
X = df.drop('song_popularity', axis=1)
y = df['song_popularity']
train_X, test_X, train_y, test_y = train_test_split(X,
                                                    y,
                                                    train_size=0.8,
                                                    test_size=0.2,
                                                    random_state=100)

# feature extraction

# check for multicollinearity
vif = pd.DataFrame()
vif["variables"] = X.columns
vif["VIF"] = [variance_inflation_factor(X, i) for i in
range(X.shape[1])]
if np.any(vif["VIF"]> 5):
    # PCA exploration
    pca = PCA().fit(train_X)
    # fig, ax = plt.subplots()
    x_values = range(1, pca.n_components_ + 1)
    # plt.bar(x_values, pca.explained_variance_ratio_, label='Explained
Variance')
    # ax.plot(x_values,
    #         np.cumsum(pca.explained_variance_ratio_),
    #         label='Cumulative Explained Variance',
    #         color='red')
    threshold = 0.95
    # plt.plot([0, pca.n_components_ + 1], [threshold, threshold], "g--
")
    f = interp1d(x_values, np.cumsum(pca.explained_variance_ratio_),
kind="linear")
    cev_x = np.linspace(1, pca.n_components_, 1000)
    cev_y = f(cev_x)  # Cumulative Explained Variance
    idx = np.argwhere(np.isclose(cev_y, [threshold] * len(cev_y),
                                 atol=1e-3)).flatten()
    idx = idx[int(len(idx) / 2)]
    # ax.scatter(cev_x[idx], cev_y[idx])
    # ax.text(cev_x[idx], cev_y[idx]-0.05,
f"({round(cev_x[idx])},{round(cev_y[idx], 2)})", fontsize=12)
    # ax.set_xlabel('Principal Component')
    # ax.set_yticks([0,0.2,0.4,0.6,0.8,1.0,threshold])
    # ax.set_xlim((0, pca.n_components_ + 1))
    # plt.legend()
    # plt.grid()
```

```
    # plt.show()

    pca = PCA(n_components=int(cev_x[idx]))
    train_X = pd.DataFrame(pca.fit_transform(train_X))
    test_X = pd.DataFrame(pca.fit_transform(test_X))

LR = LinearRegression().fit(train_X, train_y)
train_pred = LR.predict(train_X)
test_pred = LR.predict(test_X)

print('coeffecients', LR.coef_)
print('intercepts', LR.intercept_)

f, axs = plt.subplots(2, 3, sharey=True)
f.set_figwidth(15)
f.set_figheight(8)
rc = np.random.choice(range(0, pca.n_components_), len(axs[0]),
replace=False)
for i, col in enumerate(rc):
    axs[0][i].set_title(f'Training set {i+1}#')
    axs[0][i].scatter(train_X[col], train_y, label='real')
    axs[0][i].scatter(train_X[col], train_pred, label='pred')
    axs[0][i].legend()
for i, col in enumerate(rc):
    axs[1][i].set_title(f'Test set {i+1}#')
    axs[1][i].scatter(test_X[col], test_y, label='real')
    axs[1][i].scatter(test_X[col], test_pred, label='pred')
    axs[1][i].legend()

plt.show()

print('MSE of Training set', mean_squared_error(train_y, train_pred))
print('R2-Score of Testing set', r2_score(train_y, train_pred))
print('MSE of Test set', mean_squared_error(test_y, test_pred))
print('R2-Score of Testing set', r2_score(test_y, test_pred))
```