实验二

基于回归分析的大学综合得分预测

2022年6月23日

模型构建

1. 线性回归Baseline

1）划分数据集

```
RANDOM_SEED = 2020
# 划分训练集和测试集
def split(X, Y, test_size=0.2):
    x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=test_size, random_state=RANDOM_SEED)
    return x_train, x_test, y_train, y_test
```

2）构建线性回归模型

```
linreg = LinearRegression()
x_train, x_test, y_train, y_test = split(X, Y)
linreg.fit(x_train, y_train)
print(linreg.intercept_, linreg.coef_)
```

```
65.55424256838295 [-6.17588386e-02  2.82375384e-04  1.31975766e-05 -7.20591259e-03
  6.57612914e-04 -6.55744975e-03 -2.65494903e-03 -2.42756500e-03]
```

可以看到模型的截距为65.55， 各个特征的系数分别为 [-6.17588386e-02 2.82375384e-04 1.31975766e-05 -7.20591259e-03
  6.57612914e-04 -6.55744975e-03 -2.65494903e-03 -2.42756500e-03]
都是负数，因为表现越好则排名越低，是一种负相关的关系。

3）预测

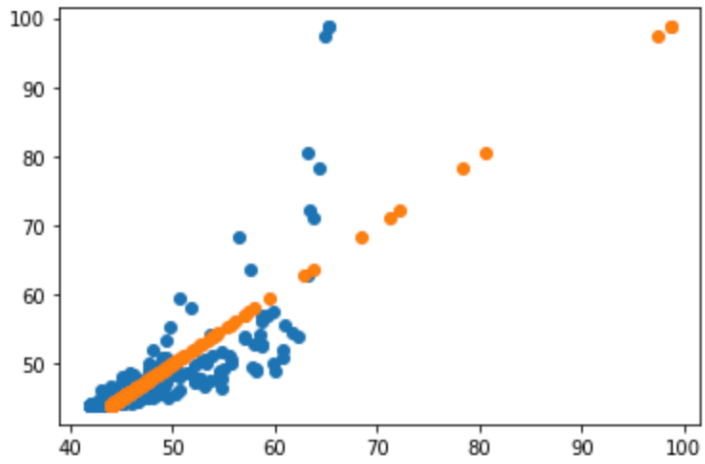
```
y_pred = linreg.predict(x_test)
print( np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
3.9990456867487096
```

预测的RMSE为3.999

4）查看拟合效果

```
plt.scatter(y_pred, y_test, marker='o')
plt.scatter(y_test, y_test)
plt.show()
```



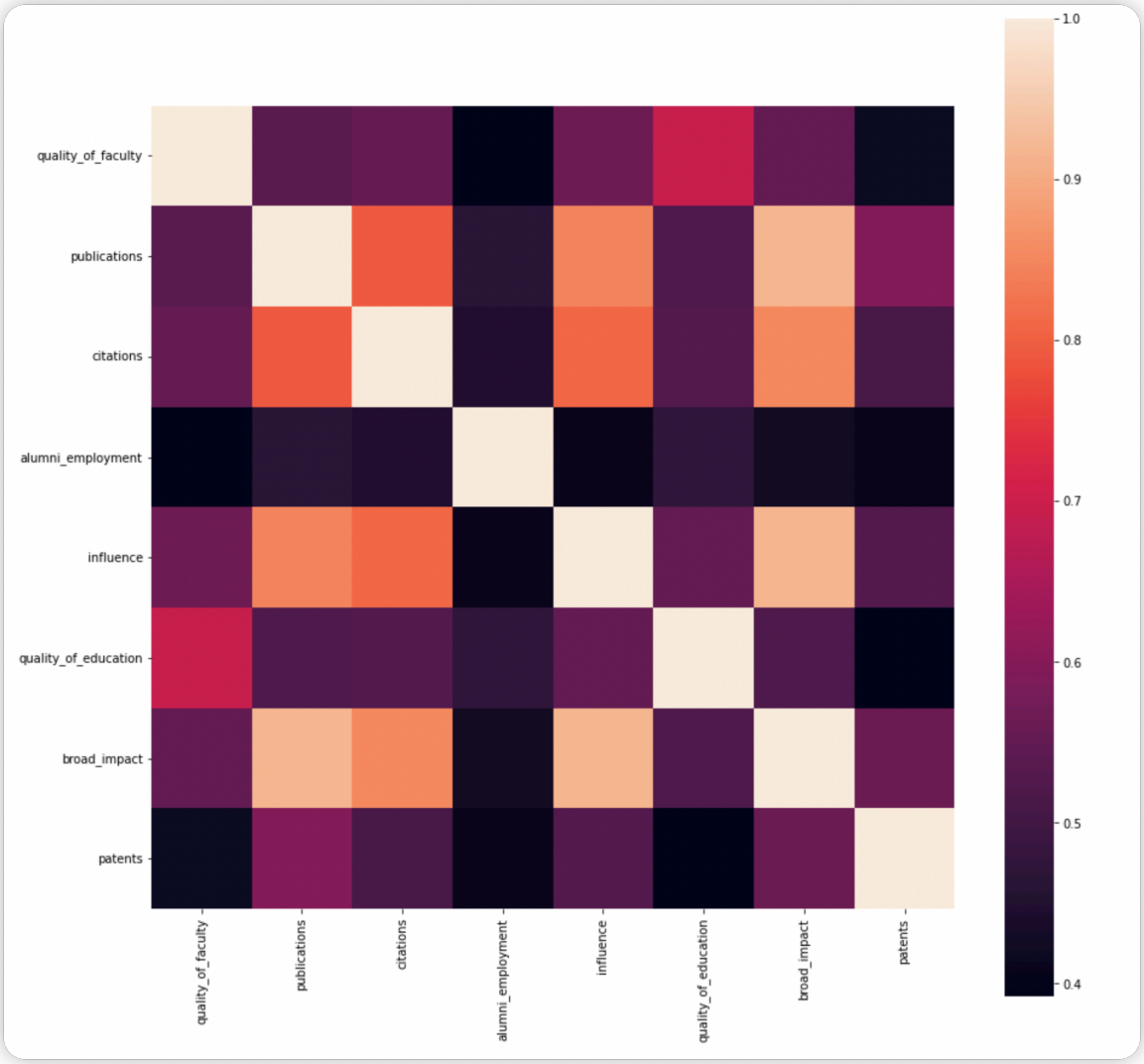
可以看出拟合效果一般，下面将从特征工程和模型调优两个方面优化

2. 特征工程

1）绘制相关系数图

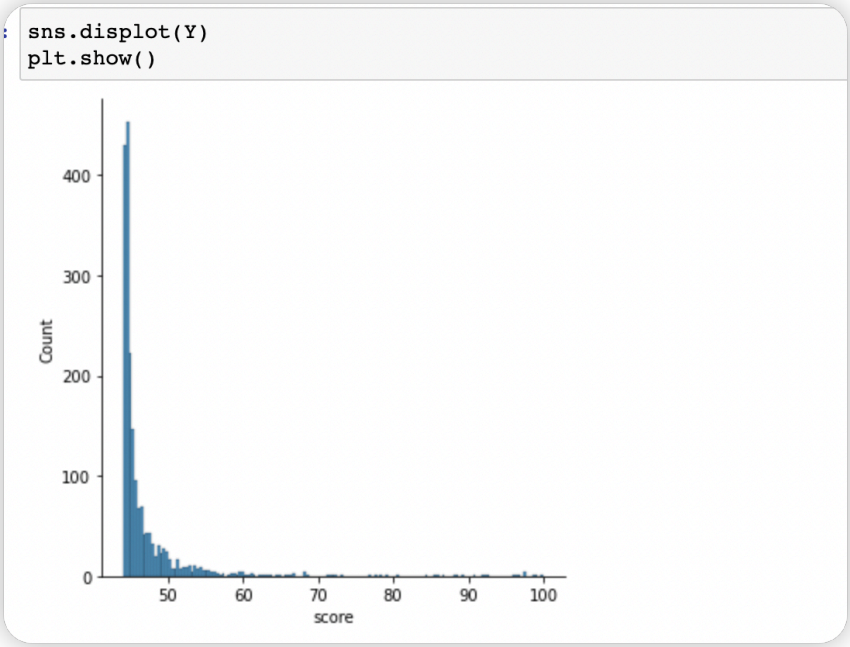
```
# 相关系数图
def draw_corr_pic(x):
    corrm = x.corr()
    plt.subplots(figsize=(15, 15))
    sns.heatmap(corrm, square=True)
    plt.show()
draw_corr_pic(X)
```

从图中可以看出 broad_impact，influence 的相关性较高，所以可以只保留其中一个特征，我这边保留influence。



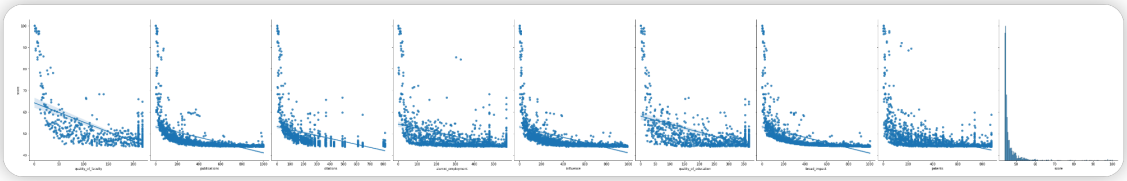
2）查看目标值分布，可以看出排名数据排后面的较多，分布呈右偏，可以采用取对数进行部分修正

```
sns.displot(Y)
plt.show()
```



3）查看特征值与目标值分布

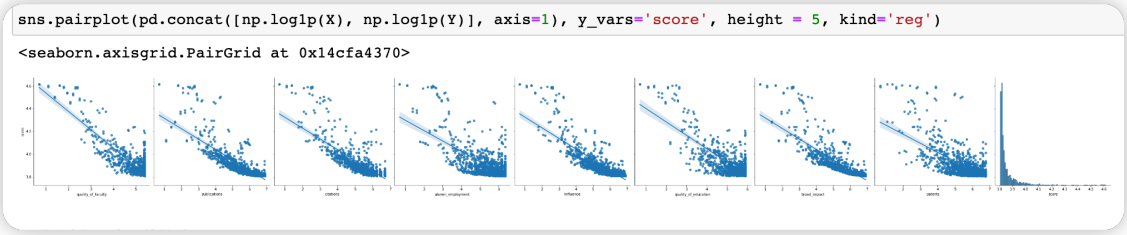
```
sns.pairplot(pd.concat([X, Y], axis=1), y_vars='score', height= 7 ,aspect = 0.8, kind='reg')
```



从数据分布在中看出X的各特征和Y线性关系都很薄弱，而线性回归的假设是两者具有线性关系。这里我采用log变换，查看变换后的是否具有线性关系。

```
sns.pairplot(pd.concat([np.log1p(X), np.log1p(Y)], axis=1), y_vars='score', height = 5, kind='reg')
```

```
<seaborn.axisgrid.PairGrid at 0x14cfa4370>
```



可以看出都具有了线性关系。

4）根据上述分析进行特征工程

```
new_X = X.copy()
```

```
new_X = np.log1p(new_X.drop(["influence"], axis =1))
new_Y = np.log1p(Y)
```

5）构建线性回归模型

```
linreg = LinearRegression()
x_train, x_test, y_train, y_test = split(new_X, new_Y)
linreg.fit(x_train, y_train)
print(linreg.intercept_, linreg.coef_)
```

```
4.7747039360744665 [-0.07388521 -0.0059417   0.00092726 -0.03081863 -0.02083327 -0.0220388
 -0.01276995]
```

调整特征后，截距和系数变小了。

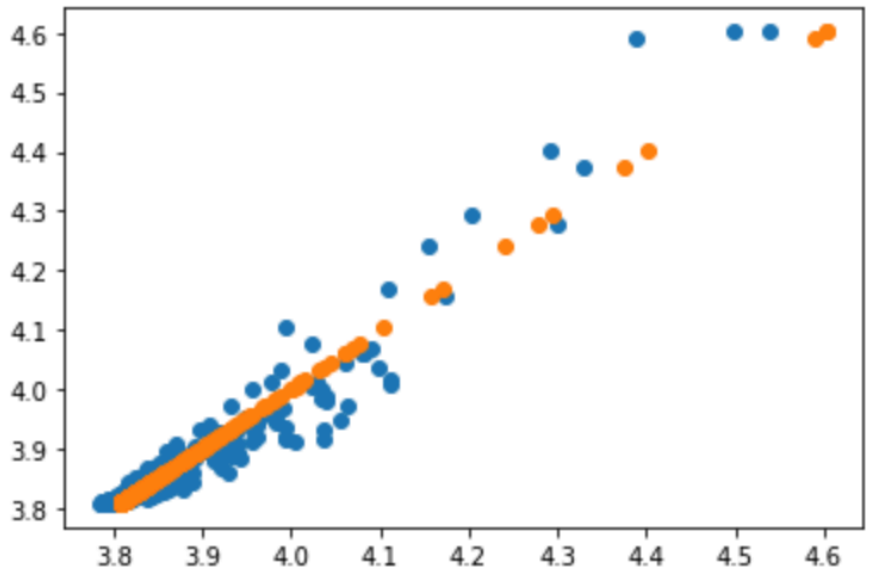
6）预测

```
y_pred = linreg.predict(x_test)
print( np.sqrt(metrics.mean_squared_error(np.exp(y_test) - 1, np.exp(y_pred) - 1)) )
```

```
1.7785045612562305
```

这里由于前面对Y进行了y = log(y+1)的计算，所以计算rmse时，需要还原Y，即y = exp(y) - 1。 计算出的RMSE为1.778，效果比最初的要好。
从分布上看，拟合度也比较高。

```
plt.scatter(y_pred, y_test, marker='o')
plt.scatter(y_test, y_test)
plt.show()
```



3. 模型调优

线性回归还提供了岭回归与Lasso回归，来解决线性回归出现的过拟合问题，这里测试了两种回归的效果。结果与普通线性回归相差不大，不过还有调参优化的空间。

```
# 岭回归
rideg = Ridge(alpha=0.0001)
rideg.fit(x_train, y_train)
y_pred = rideg.predict(x_test)
print( np.sqrt(metrics.mean_squared_error(np.exp(y_test) - 1, np.exp(y_pred) - 1)) )
```

```
1.7785046712719497
```

```
#Lasso 回归
lasso = Lasso(alpha=0.00001, fit_intercept=True)
lasso.fit(x_train, y_train)
y_pred = lasso.predict(x_test)
print( np.sqrt(metrics.mean_squared_error(np.exp(y_test) - 1, np.exp(y_pred) - 1)) )
```

```
1.7784124032883428
```

总结

本次实验使用线性回归模型来分析大学综合得分，采用 RMSE 作为评价指标。由于预测的结果是大学排名，而特征是大学各方面的表现，表现好的则排名会低，所以线性回归模型的系数都是负数。一开始直接在数据集上使用线性回归的RMSE是3.999，后面通过绘制分布图，分析得出特征和目标值并没有很好的线性关系，因此采用取对数的方法进行优化，得出的预测结果为1.778。最后采用线性回归的岭回归和Lasso回归分析，结果上看没有改善，但是还有调差优化的空间。