USTC-AD/2024 课程作业 实验报告

实验 3	威斯康辛州乳腺癌数据集探索与分析
马天开	PB21000030
Due: 2024.04.28	Submitted: 2024.04.25

Pt1: pandas

In [ ]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import scipy
```

In [ ]:
```
df = pd.read_csv("data.csv", encoding="utf-8")
df.head(10)
```

Out[ ]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	sm
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	
5	843786	M	12.45	15.70	82.57	477.1	
6	844359	M	18.25	19.98	119.60	1040.0	
7	84458202	M	13.71	20.83	90.20	577.9	
8	844981	M	13.00	21.82	87.50	519.8	
9	84501001	M	12.46	24.04	83.97	475.9	

10 rows × 32 columns

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          568 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   567 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               567 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           568 non-null    float64
 18  concavity_se             568 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     568 non-null    float64
 22  radius_worst             568 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         568 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

In [ ]:
```
# removing all rows with empty values in it:
df = df.dropna()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 560 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       560 non-null    int64  
 1   diagnosis                560 non-null    object 
 2   radius_mean              560 non-null    float64
 3   texture_mean             560 non-null    float64
 4   perimeter_mean           560 non-null    float64
 5   area_mean                560 non-null    float64
 6   smoothness_mean          560 non-null    float64
 7   compactness_mean         560 non-null    float64
 8   concavity_mean           560 non-null    float64
 9   concave points_mean      560 non-null    float64
 10  symmetry_mean            560 non-null    float64
 11  fractal_dimension_mean   560 non-null    float64
 12  radius_se                560 non-null    float64
 13  texture_se               560 non-null    float64
 14  perimeter_se             560 non-null    float64
 15  area_se                  560 non-null    float64
 16  smoothness_se            560 non-null    float64
 17  compactness_se           560 non-null    float64
 18  concavity_se             560 non-null    float64
 19  concave points_se        560 non-null    float64
 20  symmetry_se              560 non-null    float64
 21  fractal_dimension_se     560 non-null    float64
 22  radius_worst             560 non-null    float64
 23  texture_worst            560 non-null    float64
 24  perimeter_worst          560 non-null    float64
 25  area_worst               560 non-null    float64
 26  smoothness_worst         560 non-null    float64
 27  compactness_worst        560 non-null    float64
 28  concavity_worst          560 non-null    float64
 29  concave points_worst     560 non-null    float64
 30  symmetry_worst           560 non-null    float64
 31  fractal_dimension_worst  560 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 144.4+ KB
```

In [ ]:
```python
# reset df index:
df = df.reset_index(drop=True)
```

In [ ]:
```python
# dropping id column:
df = df.drop(columns=["id"])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 560 entries, 0 to 559
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   diagnosis                560 non-null    object 
 1   radius_mean              560 non-null    float64
 2   texture_mean             560 non-null    float64
 3   perimeter_mean           560 non-null    float64
 4   area_mean                560 non-null    float64
 5   smoothness_mean          560 non-null    float64
 6   compactness_mean         560 non-null    float64
 7   concavity_mean           560 non-null    float64
 8   concave points_mean      560 non-null    float64
 9   symmetry_mean            560 non-null    float64
 10  fractal_dimension_mean   560 non-null    float64
 11  radius_se                560 non-null    float64
 12  texture_se               560 non-null    float64
 13  perimeter_se             560 non-null    float64
 14  area_se                  560 non-null    float64
 15  smoothness_se            560 non-null    float64
 16  compactness_se           560 non-null    float64
 17  concavity_se             560 non-null    float64
 18  concave points_se        560 non-null    float64
 19  symmetry_se              560 non-null    float64
 20  fractal_dimension_se     560 non-null    float64
 21  radius_worst             560 non-null    float64
 22  texture_worst            560 non-null    float64
 23  perimeter_worst          560 non-null    float64
 24  area_worst               560 non-null    float64
 25  smoothness_worst         560 non-null    float64
 26  compactness_worst        560 non-null    float64
 27  concavity_worst          560 non-null    float64
 28  concave points_worst     560 non-null    float64
 29  symmetry_worst           560 non-null    float64
 30  fractal_dimension_worst  560 non-null    float64
dtypes: float64(30), object(1)
memory usage: 135.8+ KB
```

In [ ]:
```
df
```

Out[ ]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness
0	M	17.99	10.38	122.80	1001.0	
1	M	20.57	17.77	132.90	1326.0	
2	M	19.69	21.25	130.00	1203.0	
3	M	11.42	20.38	77.58	386.1	
4	M	20.29	14.34	135.10	1297.0	
...	...	...	...	...	...	
555	M	21.56	22.39	142.00	1479.0	
556	M	20.13	28.25	131.20	1261.0	
557	M	16.60	28.08	108.30	858.1	
558	M	20.60	29.33	140.10	1265.0	
559	B	7.76	24.54	47.92	181.0	

560 rows × 31 columns

In [ ]:
```
df["diagnosis"].value_counts()
```

Out[ ]:
```
diagnosis
B    355
M    205
Name: count, dtype: int64
```

In [ ]:
```
# set B -> 0, M -> 1:
df["diagnosis"] = df["diagnosis"].map({"B": 0, "M": 1})
df
```

Out[ ]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness
0	1	17.99	10.38	122.80	1001.0	
1	1	20.57	17.77	132.90	1326.0	
2	1	19.69	21.25	130.00	1203.0	
3	1	11.42	20.38	77.58	386.1	
4	1	20.29	14.34	135.10	1297.0	
...	...	...	...	...	...	
555	1	21.56	22.39	142.00	1479.0	
556	1	20.13	28.25	131.20	1261.0	
557	1	16.60	28.08	108.30	858.1	
558	1	20.60	29.33	140.10	1265.0	
559	0	7.76	24.54	47.92	181.0	

560 rows × 31 columns

In [ ]:
```python
def handle(column: str) -> None:
    # for each colum, print mid, var, q1 q3, min, max, mean, median etc...:
    print(f"Column: {column}")
    print(f"Mean: {df[column].mean()}")
    # print(f"Median: {df[column].median()}")
    # print(f"Variance: {df[column].var()}")
    print(f"Standard Deviation: {df[column].std()}")
    print(f"Minimum: {df[column].min()}")
    print(f"Maximum: {df[column].max()}")
    print(f"Q1: {df[column].quantile(0.25)}")
    print(f"Q3: {df[column].quantile(0.75)}")
    print()

_ = [handle(column) for column in ["radius_mean", "texture_mean", "perimeter
```

```
Column: radius_mean
Mean: 14.074301785714287
Standard Deviation: 3.491064449570111
Minimum: 6.981
Maximum: 28.11
Q1: 11.6775
Q3: 15.75

Column: texture_mean
Mean: 19.27175
Standard Deviation: 4.319014680293036
Minimum: 9.71
Maximum: 39.28
Q1: 16.1575
Q3: 21.802500000000002

Column: perimeter_mean
Mean: 91.59585714285713
Standard Deviation: 24.048328628379842
Minimum: 43.79
Maximum: 188.5
Q1: 74.9675
Q3: 103.725

Column: area_mean
Mean: 649.6439285714287
Standard Deviation: 347.45128727529766
Minimum: 143.5
Maximum: 2501.0
Q1: 418.325
Q3: 775.775

Column: smoothness_mean
Mean: 0.09628135714285714
Standard Deviation: 0.014087910812849068
Minimum: 0.05263
Maximum: 0.1634
Q1: 0.08629
Q3: 0.1051
```

In [ ]:
```python
# group by diagnosis, calculate CV for each column:
df_grouped = df.groupby("diagnosis")
df_grouped.std() / df_grouped.mean()
```

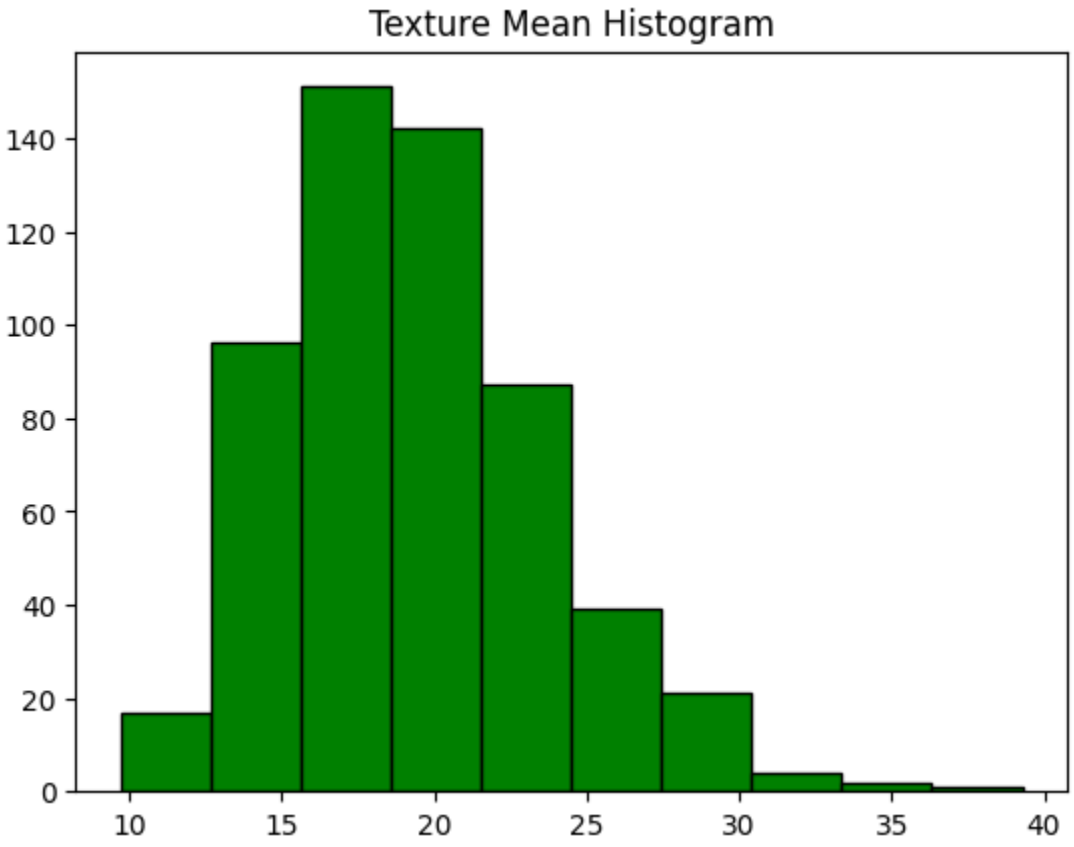
Out[ ]:

diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
0	0.146810	0.223512	0.151502	0.290776	0.14549
1	0.182084	0.175314	0.187713	0.373423	0.12403

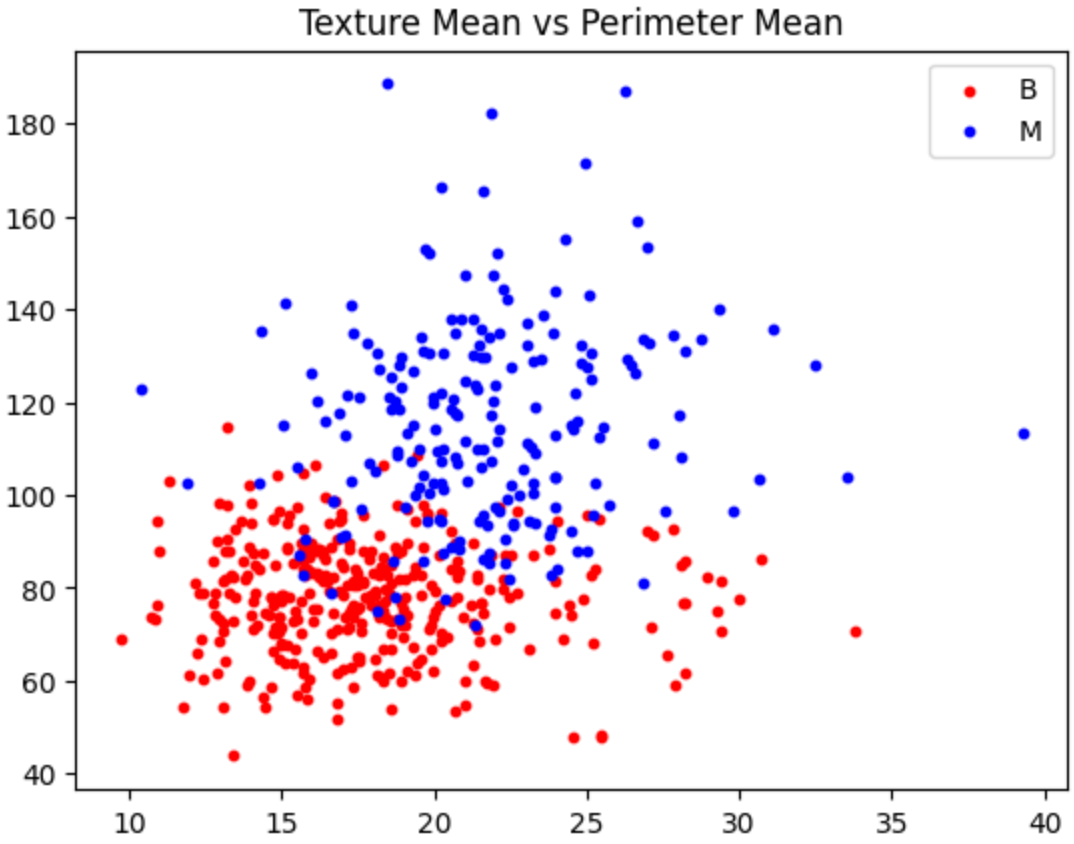
2 rows × 30 columns

Pt2: matplotlib

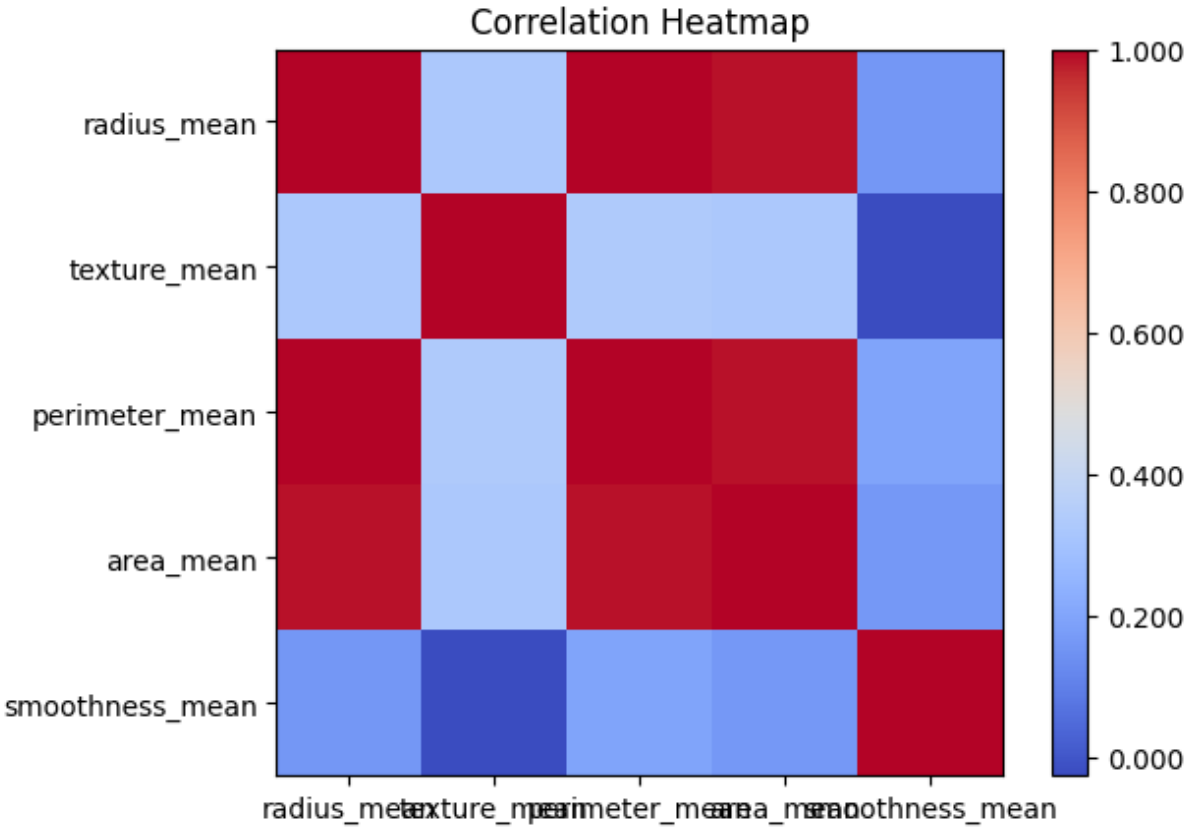
In [ ]:
```
_ = plt.hist(df["texture_mean"], bins=10, color="green", edgecolor="black")
_ = plt.title("Texture Mean Histogram")
```



In [ ]:
```
df_0 = df[df["diagnosis"] == 0]
df_1 = df[df["diagnosis"] == 1]
_ = plt.scatter(df_0["texture_mean"], df_0["perimeter_mean"], c="red", label
_ = plt.scatter(df_1["texture_mean"], df_1["perimeter_mean"], c="blue", labe
plt.legend()
_ = plt.title("Texture Mean vs Perimeter Mean")
```



In [ ]:
```
columns = ["radius_mean", "texture_mean", "perimeter_mean", "area_mean", "sm
corr = df[columns].corr()
_ = plt.imshow(corr, cmap="coolwarm", interpolation="nearest")
_ = plt.colorbar(format="%.3f")
_ = plt.xticks(range(len(columns)), columns)
_ = plt.yticks(range(len(columns)), columns)
_ = plt.title("Correlation Heatmap")
```



Pt3: 线性回归

In [ ]:
```
X = df["radius_mean"]
Y = df["area_mean"]
X2 = X ** 2
X = np.column_stack((X, X2))
X = np.column_stack((np.ones(X.shape[0]), X))
W1 = np.linalg.inv(X.T @ X) @ X.T @ Y
W1
```

Out[ ]:
```
array([-4.70867951, -0.44260792,  3.14186228])
```

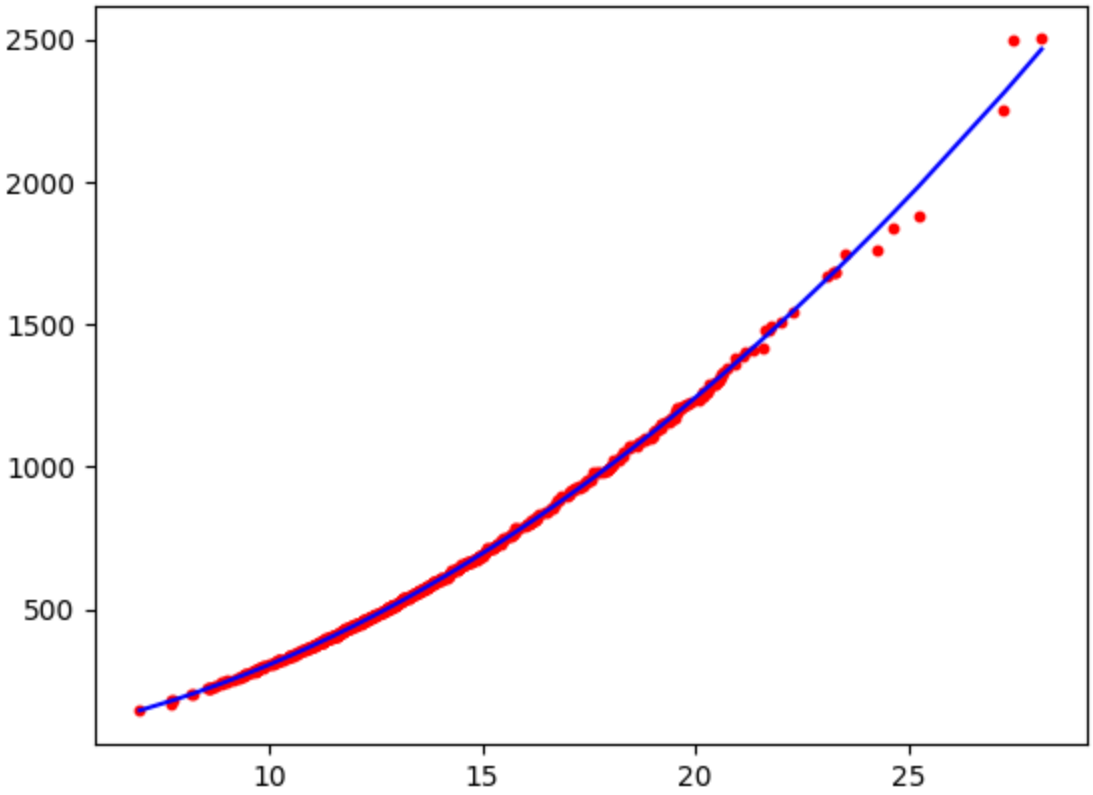
In [ ]:
```
W2 = np.polyfit(df["radius_mean"], df["area_mean"], 2)
W2
```

Out[ ]:
```
array([ 3.14186228, -0.44260792, -4.70867951])
```

In [ ]:
```
X = df["radius_mean"]
Y = df["area_mean"]
X = np.sort(X)
Y = np.sort(Y)
_ = plt.scatter(X, Y, s=10, color="red")
_ = plt.plot(X, W1[0] + W1[1] * X + W1[2] * X ** 2, color="blue")
_ = plt.label("Linear Regression")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[17], line 7
      5 _ = plt.scatter(X, Y, s=10, color="red")
      6 _ = plt.plot(X, W1[0] + W1[1] * X + W1[2] * X ** 2, color="blue")
----> 7 _ = plt.label("Linear Regression")

AttributeError: module 'matplotlib.pyplot' has no attribute 'label'
```



从散点图可以看出 radius_mean 与 area_mean 之间有较强的相关性, 线性回归能很好的拟合这两者之间的关系

Pt4: 数据降维

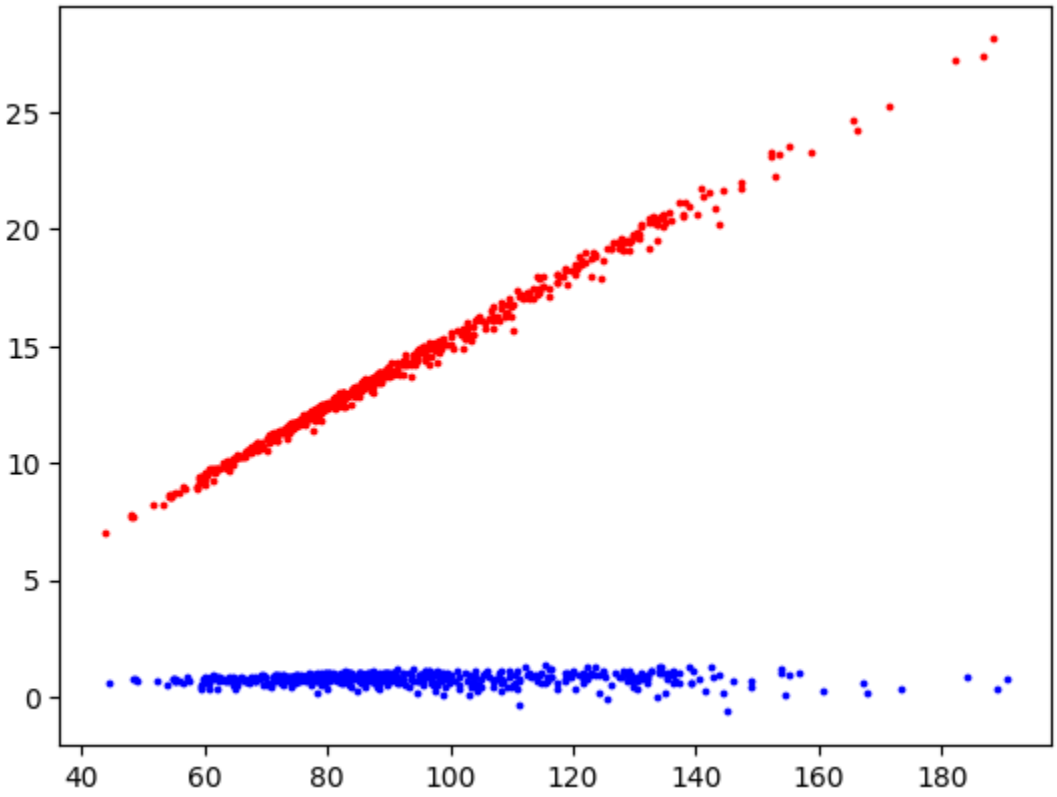
我们在下面简要讨论 PCA Principle component analysis 主成分分析的简要思路.

因为没读懂 exp3.pdf 中提示的思路, 我们直接提供从 X 到 new_X (Z) 的代码

In [ ]:
```
X = df[["perimeter_mean", "radius_mean"]].to_numpy()
mean_vec= np.mean(X, axis=0)
cov_mat = np.cov(X - mean_vec, rowvar=False)
fvalue, fvec = np.linalg.eig(cov_mat)
fvalue_sort = np.argsort(fvalue)[::-1]
fvalue = fvalue[fvalue_sort]
fvec = fvec[:, fvalue_sort]
new_X = X @ fvec
print(new_X)
```

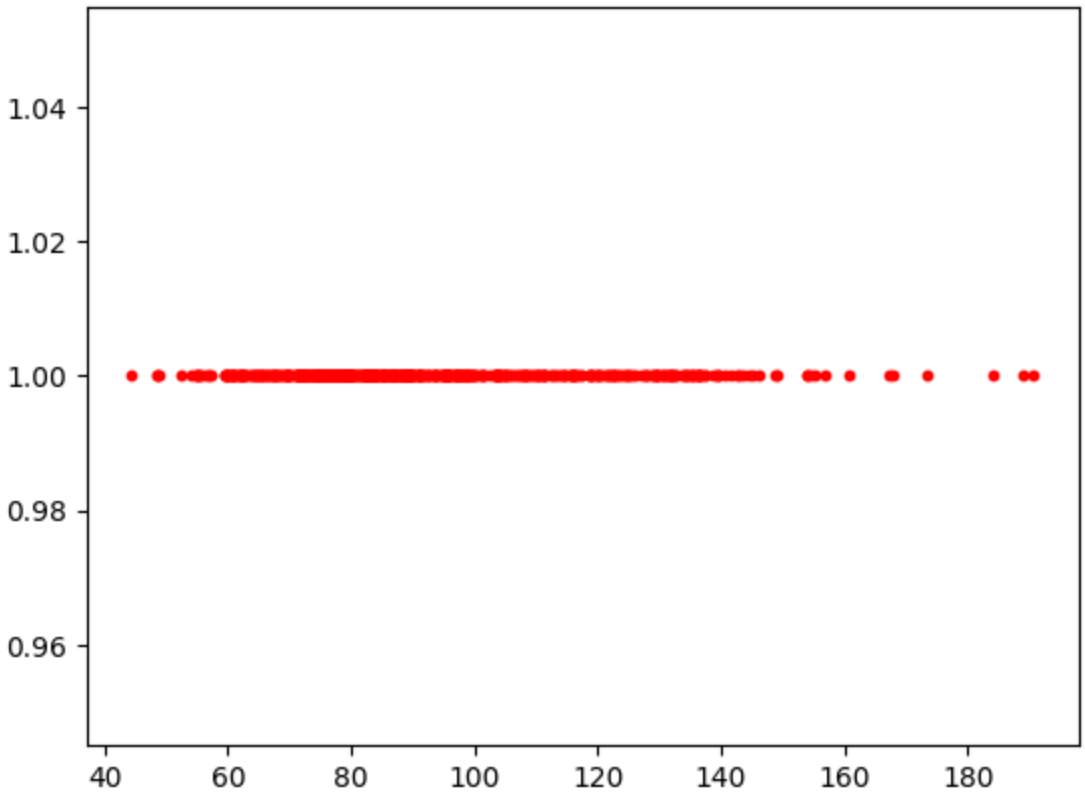
```
[[124.11059852   0.19858123]
 [134.47614923   1.30391313]
 [131.47994405   0.84877078]
 ...
 [109.5611099    0.90177456]
 [141.60607044   0.30135467]
 [ 48.53749552   0.8096475 ]]
```

In [ ]:
```
_ = plt.scatter(X[:, 0], X[:, 1], s=3, color="red")
_ = plt.scatter(new_X[:, 0], new_X[:, 1], s=3, color="blue")
```



从图中可以看到 new_X 的 y 的分布集中在 0 附近, 我们可以舍弃 new_X 中的 y 以降低数据维度

In [ ]:
```
X_pca = new_X[:, 0]
_ = plt.scatter(X_pca, np.ones(X_pca.shape[0]), s=10, color="red")
```



Pt5: T检验

In [ ]:
```
df_grouped = df.groupby("diagnosis")
concavity_worst = df_grouped["concavity_worst"].apply(list)
print(concavity_worst)
```

```
diagnosis
0    [0.239, 0.189, 0.08867, 0.04833, 0.0688, 0.305...
1    [0.7119, 0.2416, 0.4504, 0.6869, 0.4, 0.5355, ...
Name: concavity_worst, dtype: object
```

Q1: 简述本情境下应使用成组检验还是成对检验?

A1: 本情境下应使用成对检验, 因为我们要比较的是两个不同的数据集的均值, 而不是同一个数据集的两个不同的样本的均值

Q2: 计算两组数据的平均值, 写出探测检验原假设:

In [ ]:
```
B = concavity_worst[0]
M = concavity_worst[1]
B_mean = np.mean(B)
M_mean = np.mean(M)
print(B_mean, M_mean)
```

```
0.1663615971830986 0.44671356097560977
```

单侧检验: $H_0 : \mu_1 - \mu_2 = 0$ vs $H_1 : \mu_1 - \mu_2 > 0$

Q3: 使用 `scipy.stats` 中的相关方法, 执行量样本单侧T检验:

In [ ]:
```
t, p = scipy.stats.ttest_ind(B, M)
print(t, p)
```

```
-20.346631967479436 2.928169352602799e-69
```

Q4: 简述你从以上两样本 T 检验的结果中能得出什么结论?

A4: $t = -20.3, p = 2.92e-69$, 因为 $p < 0.05$, 我们拒绝原假设, 即两组数据的均值不相等

In [ ]: