مجید ادیبیان 9631003 HW2 گزارش پیاده سازی

قسمت اول:

1. در ابتدا آماده سازی داده ها را انجام می دهیم. با بررسی هایی که بر روی داده صورت گرفته مشخص شده است که مقادیر nan به معنی missing value در داده ها وجود دارد که باید حذف شود. ولی با حذف سطر های دارای مقدار nan سطرهای کمی باقی می ماند. با بررسی که صورت گرفته متوجه شدیم که بیشتر مقادیر nan در ستون Cabin وجود دارد و 687 سطر از 891 تعداد سطر موجود مقدار Cabin=nan دارند. پس حذف این ستون مناسب به نظر می رسد. با حذف این ستون و سپس حذف سطر های دارد مقدار nan ، تعداد 712 سطر باقی می ماند.

برای استفاده از الگوریتم درخت تصمیم باید مقادیر غیر عددی را به عددی تبدیل کنیم. بدین منظور تابع convert_non_numerical ساخته این که این کار را انجام می دهد. در این تابع روی ستون‌های داده ها پیمایش می کنیم و اگر type آن ها غیر عددی است (غیر float و int) مقادیر یونیک آن ستون را استخراج می کنیم و یک دیکشنری از مقادیر آن ستون به اعدادی از صفر تا تعداد مقادیر یونیک را می سازیم و به جای هر مقدار در آن ستون value آن در دیکشنری را جایگزین می‌کنیم.

خروجی این تابع بر روی داده ها به صورت زیر است که می بینیم همه مقادیر عددی هستند:

```
<bound method NDFrame.head of             Pclass  Name  Sex   Age  SibSp  Parch  Ticket     Fare  Embarked
PassengerId
1                 3   204    0  22.0      1      0       9   7.2500         1
2                 1   655    1  38.0      1      0     163  71.2833         2
3                 3    96    1  26.0      0      0     305   7.9250         1
4                 1   412    1  35.0      1      0     217  53.1000         1
5                 3   189    0  35.0      0      0     385   8.0500         1
...             ...   ...  ...   ...    ...    ...     ...      ...       ...
886               3   620    1  39.0      0      5     346  29.1250         0
887               2   368    0  27.0      0      0     371  13.0000         1
888               1   624    1  19.0      0      0      70  30.0000         1
890               1    39    0  26.0      0      0      31  30.0000         2
891               3    60    0  32.0      0      0     505   7.7500         0

[712 rows x 9 columns]>
```

2. در مرحله بعدی الگوریتم درخت تصمیم را بر روی این داده های بدست آمده اجرا می کنیم. برای این منظور از کتابخانه sklearn استفاده کرده ایم. دقت و نمایش درخت را برای دو تابع آنتروپی و gini و برای عمق های مختلف بدست آورده ایم که خروجی ها به صورت زیر است:

entropy: برای عمق های 1 تا 10 درخت تصمیم را ساخته ایم و در هر مرحله دقت را بدست آورده ایم و در انتها بهترین دقت را چاپ کرده و درخت آن را رسم کرده ایم:

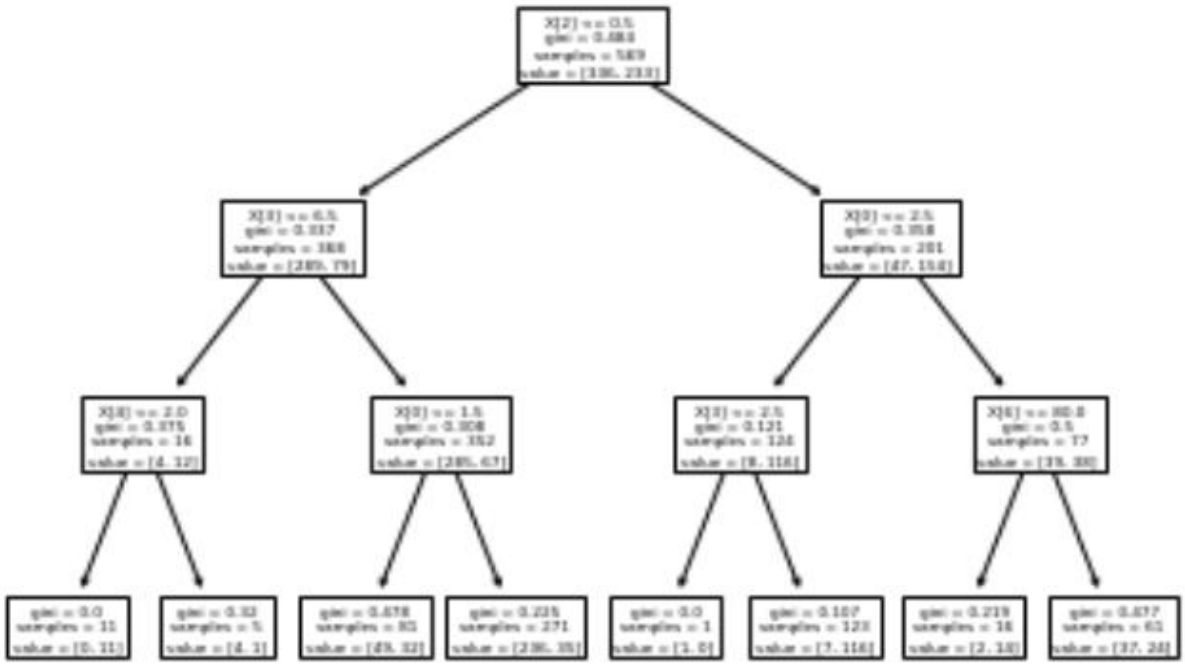
```
Accuracy for depth=1: 0.7832167832167832
Accuracy for depth=2: 0.8321678321678322
Accuracy for depth=3: 0.8531468531468531
Accuracy for depth=4: 0.8041958041958042
Accuracy for depth=5: 0.8181818181818182
Accuracy for depth=6: 0.8251748251748252
Accuracy for depth=7: 0.8111888111888111
Accuracy for depth=8: 0.8041958041958042
Accuracy for depth=9: 0.7902097902097902
Accuracy for depth=10: 0.8041958041958042

Best tree: depth=3   accuracy=0.8531468531468531
```

Gini: مانند قسمت قبل برای عمق های 1 تا 10 درخت تصمیم را ساخته ایم و در هر مرحله دقت را بدست آورده ایم و در انتها بهترین دقت را چاپ کرده و درخت آن را رسم کرده ایم:

```
Accuracy for depth=1: 0.7832167832167832
Accuracy for depth=2: 0.8321678321678322
Accuracy for depth=3: 0.8531468531468531
Accuracy for depth=4: 0.8041958041958042
Accuracy for depth=5: 0.8321678321678322
Accuracy for depth=6: 0.8251748251748252
Accuracy for depth=7: 0.8111888111888111
Accuracy for depth=8: 0.8041958041958042
Accuracy for depth=9: 0.7902097902097902
Accuracy for depth=10: 0.7902097902097902

Best tree: depth=3   accuracy=0.8531468531468531
```

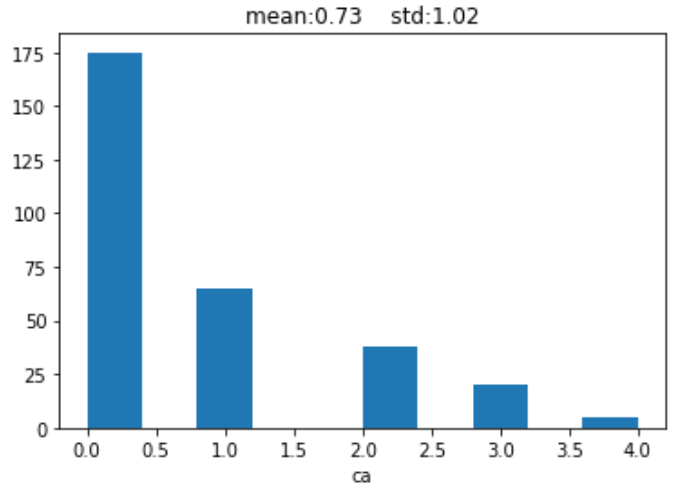


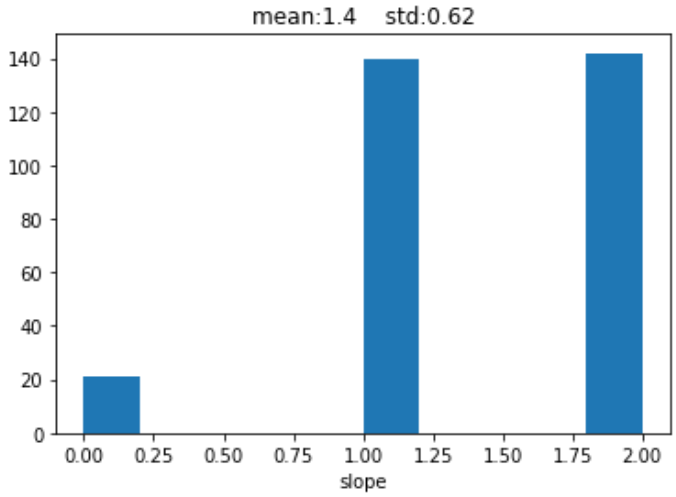
قسمت دوم:

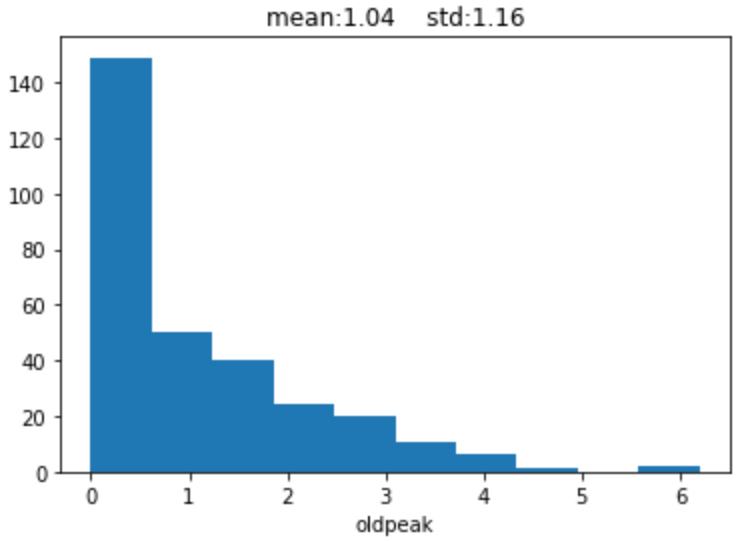
1. بررسی های آماری ویژگی ها و ساخت ویژگی جدید:

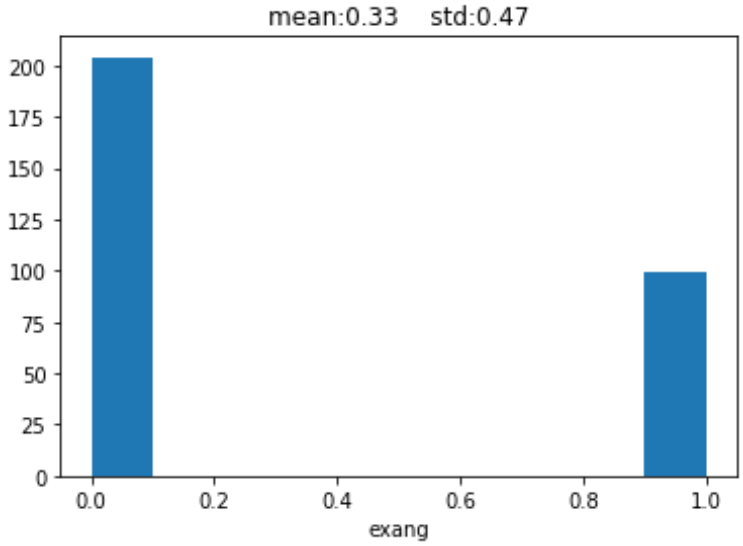
ابتدا با برررسی داده ها بررسی می کنیم که مقدار nan در داده ها وجود نداشته باشد که مشخص می شود تمام سطر ها مقدار دارند.

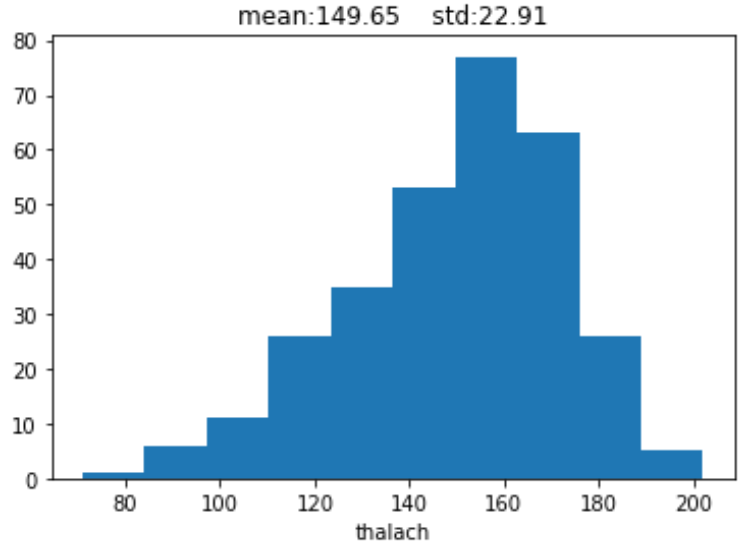
سپس داده های هر ستون را در نمودار هیستوگرام رسم کرده ایم و مقدار میانگین و انحراف معیار هر کدام را بر روی نمودار آن نوشته ایم تا دید آماری از داده ها را به ما بدهد. خروجی به صورت زیر است:

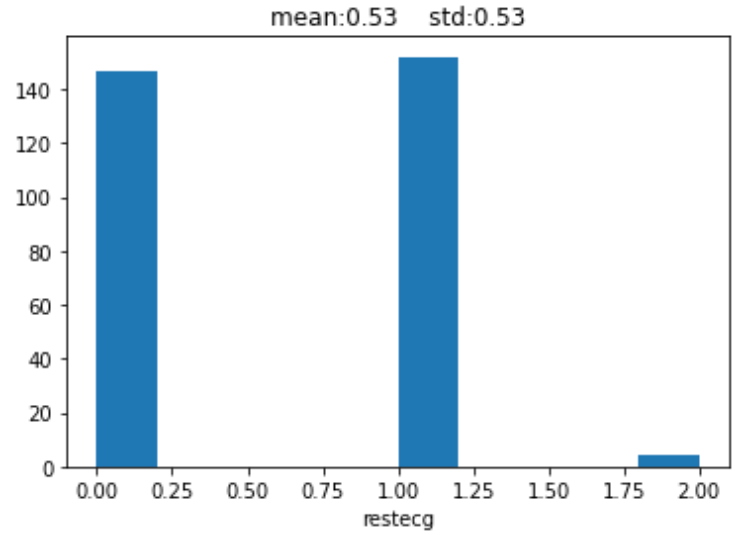


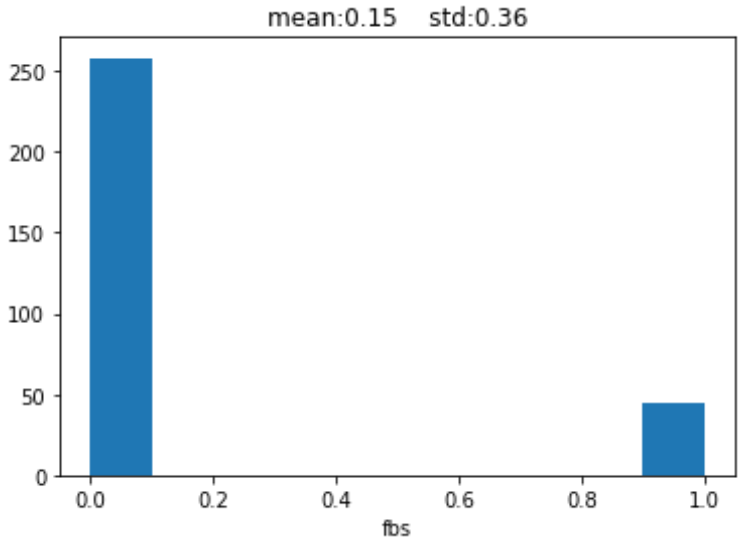


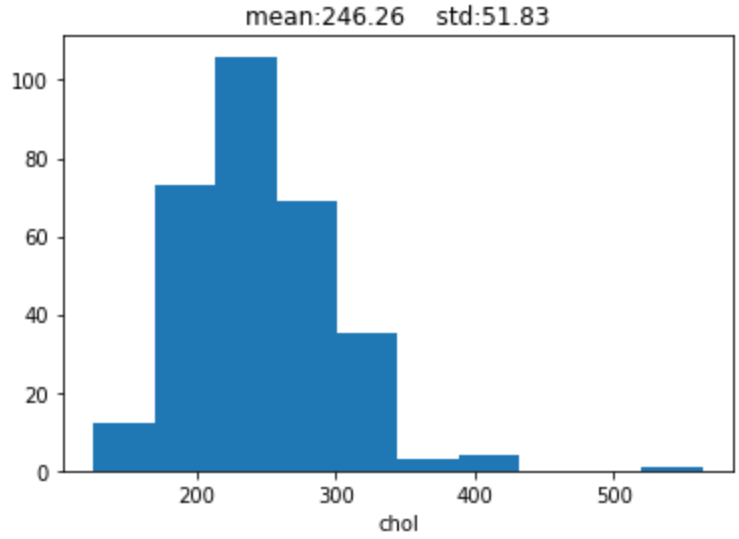


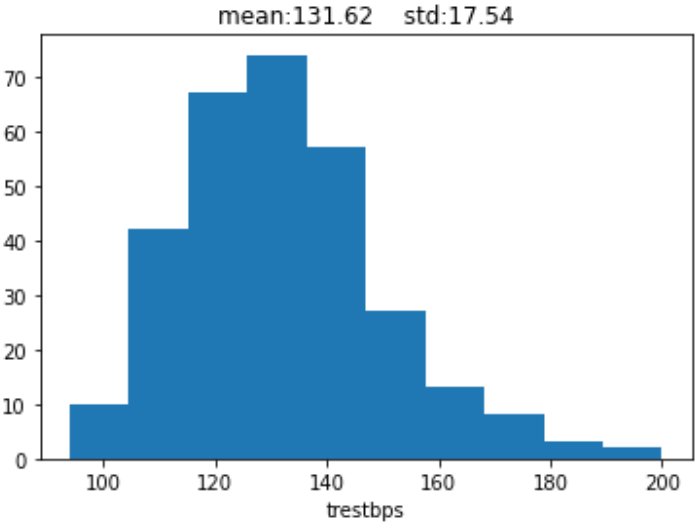

mean:0.33 std:0.47
200
175
150
125
100
75
50
25
0
0.0
0.2
0.4
0.6
0.8
1.0
exang

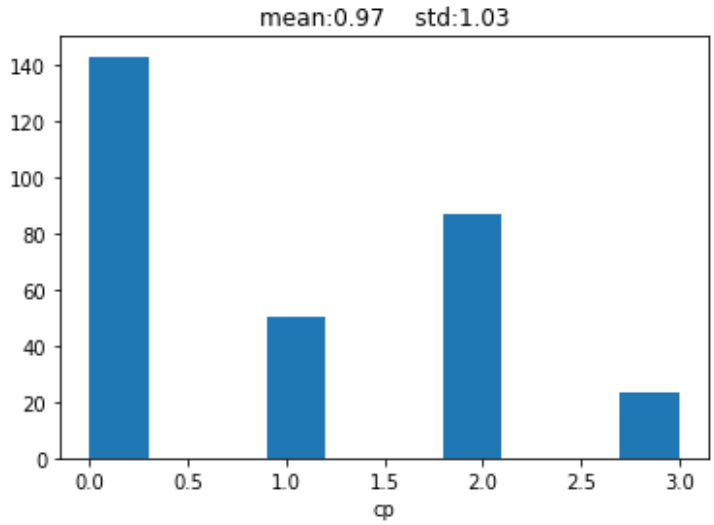

mean:149.65 std:22.91
80
70
60
50
40
30
20
10
0
80
100
120
140
160
180
200
thalach

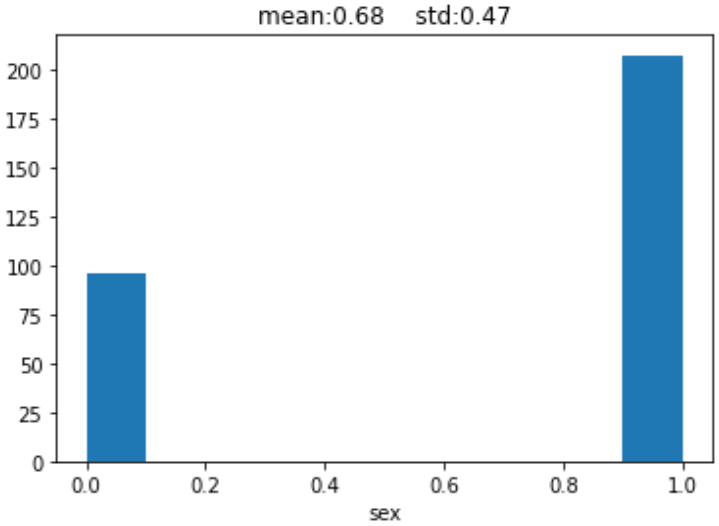

mean:0.53 std:0.53
140
120
100
80
60
40
20
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
restecg

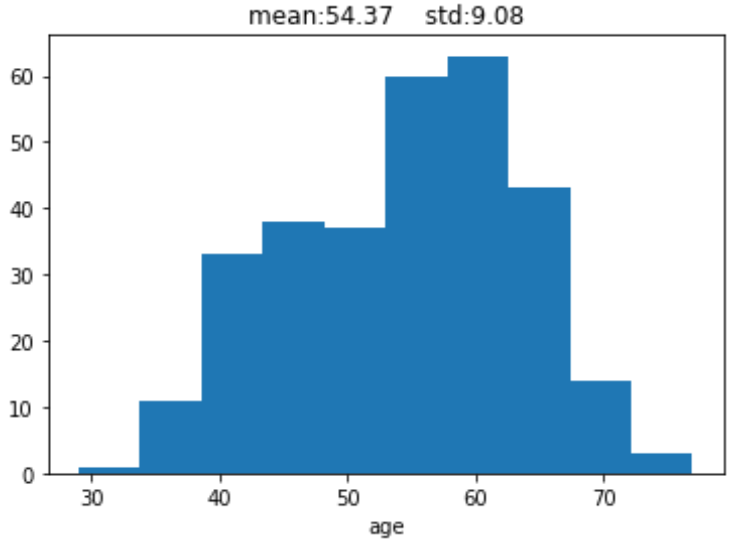

mean:0.15 std:0.36
250
200
150
100
50
0
0.0
0.2
0.4
0.6
0.8
1.0
fbs


mean:246.26 std:51.83
100
80
60
40
20
0
200
300
400
500
chol

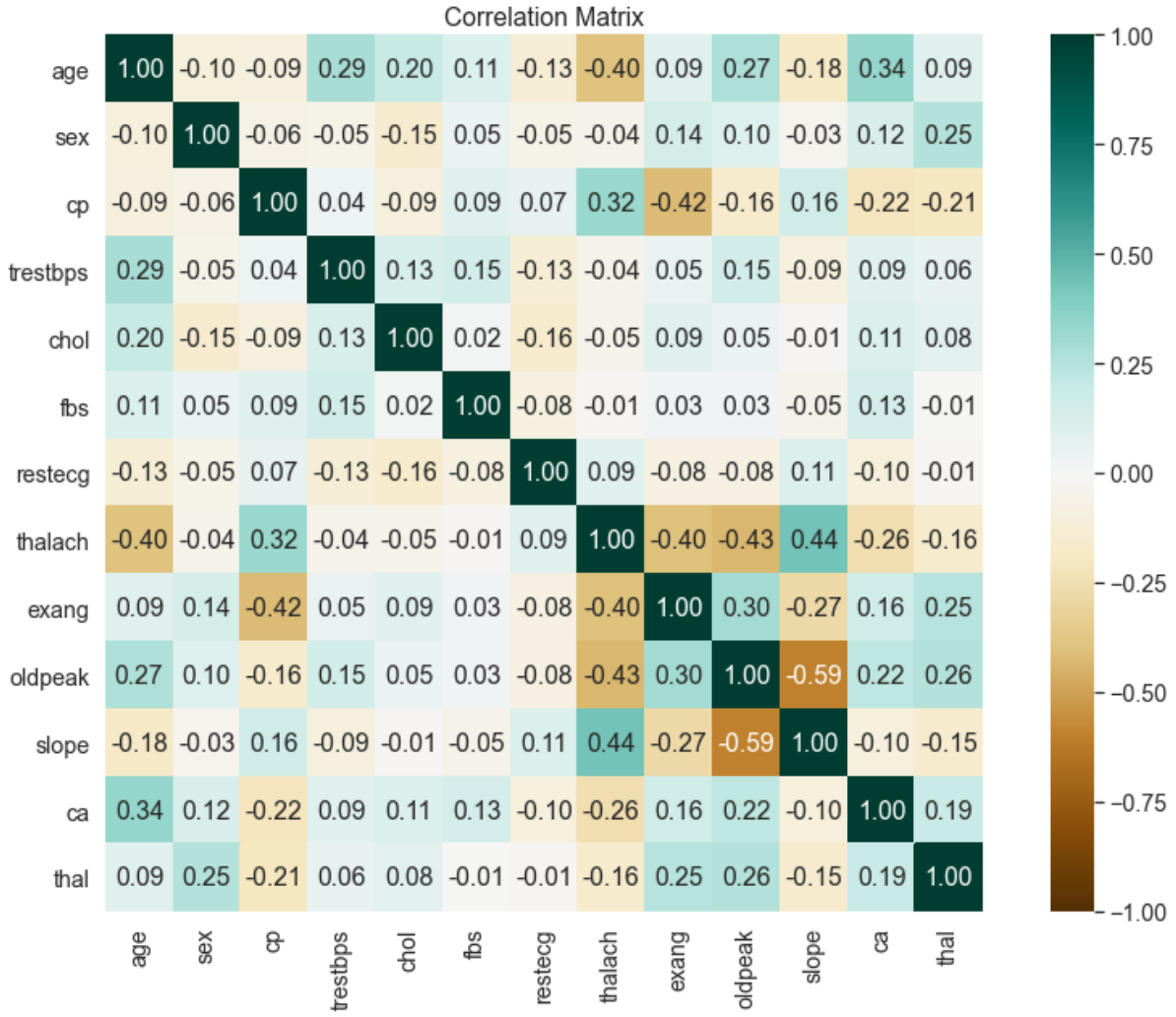

mean:131.62 std:17.54
70
60
50
40
30
20
10
0
100
120
140
160
180
200
trestbps


mean:0.97 std:1.03
140
120
100
80
60
40
20
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
cp


mean:0.68 std:0.47
200
175
150
125
100
75
50
25
0
0.0
0.2
0.4
0.6
0.8
1.0
sex


mean:54.37 std:9.08
60
50
40
30
20
10
0
30
40
50
60
70
age

در مرحله بعد مقدار correlation بین ویژگی ها را بررسی می کنیم. بدین منظور ابتدا مقدار داده های هر ستون را نرمال می کنیم و به مقداری بین 0 تا 1 تناظر می دهیم و سپس ماتریس correlation آن را محاسبه می کنیم و به صورت زیر نمایش می دهیم:



دیده می شود که بعضی از ویژگی ها ارتباط زیادی با یک دیگر دارند. ویژگی هایی که مقدار ارتباط داخل ماتریس برای آن ها بیشتر از 0.4 است و 1 نیست را جدا می کنیم. این ویژگی ها ('slope', 'thalach') و ('thalach', 'slope') هستند. با ترکیب این ویژگی ها ستون های جدیدی برای این داده ها می سازیم.

خروجی داده های جدید به صورت زیر خواهد بود:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	slope_thalach	thalach_slope
0	0.818182	1.0	1.000000	0.725	0.413121	1.0	0.0	0.742574	0.0	0.370968	0.0	0.00	0.333333	0.371287	0.371287
1	0.480519	1.0	0.666667	0.650	0.443262	0.0	0.5	0.925743	0.0	0.564516	0.0	0.00	0.666667	0.462871	0.462871
2	0.532468	0.0	0.333333	0.650	0.361702	0.0	0.0	0.851485	0.0	0.225806	1.0	0.00	0.666667	0.925743	0.925743
3	0.727273	1.0	0.333333	0.600	0.418440	0.0	0.5	0.881188	0.0	0.129032	1.0	0.00	0.666667	0.940594	0.940594
4	0.740260	0.0	0.000000	0.600	0.627660	0.0	0.5	0.806931	1.0	0.096774	1.0	0.00	0.666667	0.903465	0.903465
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	0.740260	0.0	0.000000	0.700	0.427305	0.0	0.5	0.608911	1.0	0.032258	0.5	0.00	1.000000	0.554455	0.554455
299	0.584416	1.0	1.000000	0.550	0.468085	0.0	0.5	0.653465	0.0	0.193548	0.5	0.00	1.000000	0.576733	0.576733
300	0.883117	1.0	0.000000	0.720	0.342199	1.0	0.5	0.698020	0.0	0.548387	0.5	0.50	1.000000	0.599010	0.599010
301	0.740260	1.0	0.000000	0.650	0.232270	0.0	0.5	0.569307	1.0	0.193548	0.5	0.25	1.000000	0.534653	0.534653
302	0.740260	0.0	0.333333	0.650	0.418440	0.0	0.0	0.861386	0.0	0.000000	0.5	0.25	0.666667	0.680693	0.680693

در انتها داده ها را به نسبت 80 به 20 به دو قسمت آموزشی و تست تقسیم می کنیم.

2. در پیاده سازی الگوریتم های naïve Bayes و KNN از کتابخانه sklearn استفاده شده است. در الگوریتم بیز از GaussianNB استفاده شده است و پس از آموزش مدل بر روی داده های آموزشی، دقت بدست آمده بر روی داده های تست حدود 80 درصد است:

```
accurecy in naive bayes: 0.8360655737704918
```

در الگوریتم KNN از KNeighborsClassifier استفاده شده است و آموزش بر روی داده های آموزش و دقت بر روی داده های تست برای k های مختلف انجام می شود و بهترین k که بیشترین دقت را به ما می دهد را در انتها چاپ می کنیم. یک نمونه از خروجی اجرا به صورت زیر است که در آن بهترین دقت به ازای k=9 بدست امده و حدود 70 درصد است:

```
accurecy in knn with k=1: 0.7540983606557377
accurecy in knn with k=2: 0.7049180327868853
accurecy in knn with k=3: 0.8032786885245902
accurecy in knn with k=4: 0.7704918032786885
accurecy in knn with k=5: 0.8360655737704918
accurecy in knn with k=6: 0.7868852459016393
accurecy in knn with k=7: 0.7868852459016393
accurecy in knn with k=8: 0.7540983606557377
accurecy in knn with k=9: 0.7868852459016393
accurecy in knn with k=10: 0.7704918032786885
accurecy in knn with k=11: 0.8032786885245902
accurecy in knn with k=12: 0.8032786885245902
accurecy in knn with k=13: 0.819672131147541
accurecy in knn with k=14: 0.8032786885245902
accurecy in knn with k=15: 0.8032786885245902
accurecy in knn with k=16: 0.7868852459016393
accurecy in knn with k=17: 0.8032786885245902
accurecy in knn with k=18: 0.7868852459016393
accurecy in knn with k=19: 0.7868852459016393
```

```
accurecy in knn with k=20: 0.8032786885245902
#####################################################
Best accurecy in knn is with k=5: 0.8360655737704918
#####################################################
```