باسمه تعالی

تکلیف سری سوم داده کاوی

سارا برادران (شماره دانشجویی : ۹۶۲۴۱۹۳)

سوال ۱)

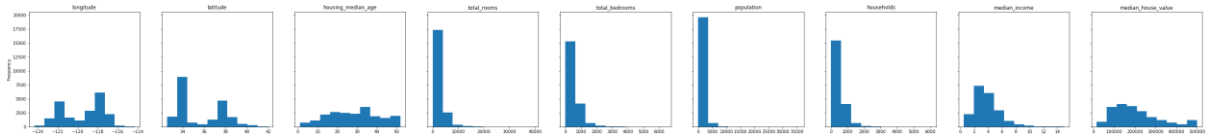
قسمت a) با استفاده از متد read_csv کتابخانه pandas داده ها را خوانده و با استفاده از متد dropna() رکورد هایی از این دیتاست که حاوی مقادیر null یا همان NaN هستند را حذف می کنیم.

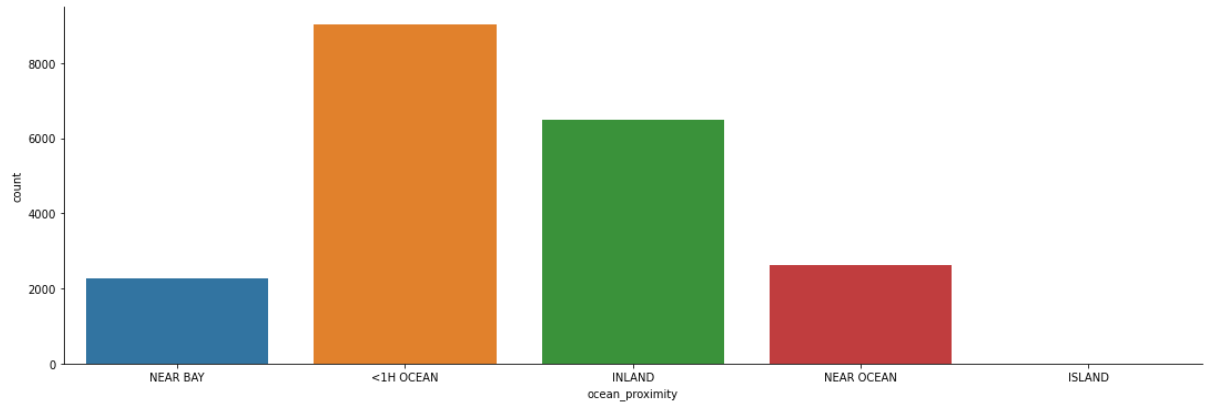
قسمت b) با فراخوانی متد value_counts() برای یک series تمام مقادیر یکتای موجود در ان series به همراه تعداد دفعات تکرار هر مقدار در داخل داده ها بدست خواهد آمد.

```
In [3]:
1  # 1_b
2
3  df['ocean_proximity'].value_counts()
```

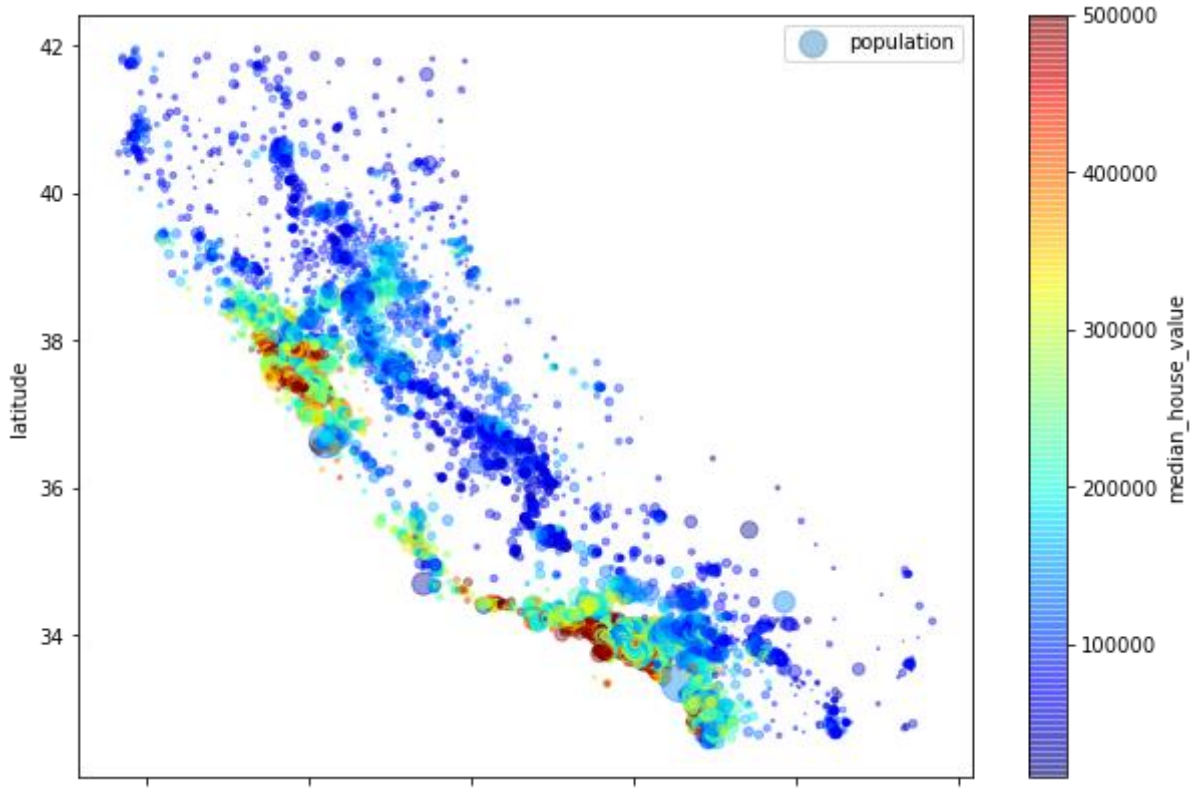
```
Out[3]: <1H OCEAN     9034
        INLAND        6496
        NEAR OCEAN    2628
        NEAR BAY      2270
        ISLAND           5
        Name: ocean_proximity, dtype: int64
```

قسمت c) دیتاست housing دارای ۹ ستون از نوع float بوده و ستون آخر و دهم آن از نوع object می باشد. نمودار هیستوگرام برای ستون های از نوع object قابل رسم نمی باشد لذا برای این ستون نمودار bar chart رسم خواهیم کرد. تمام نمودار های مربوطه به کمک یک حلقه بر روی تمام ستون های دیتاست رسم می شوند.





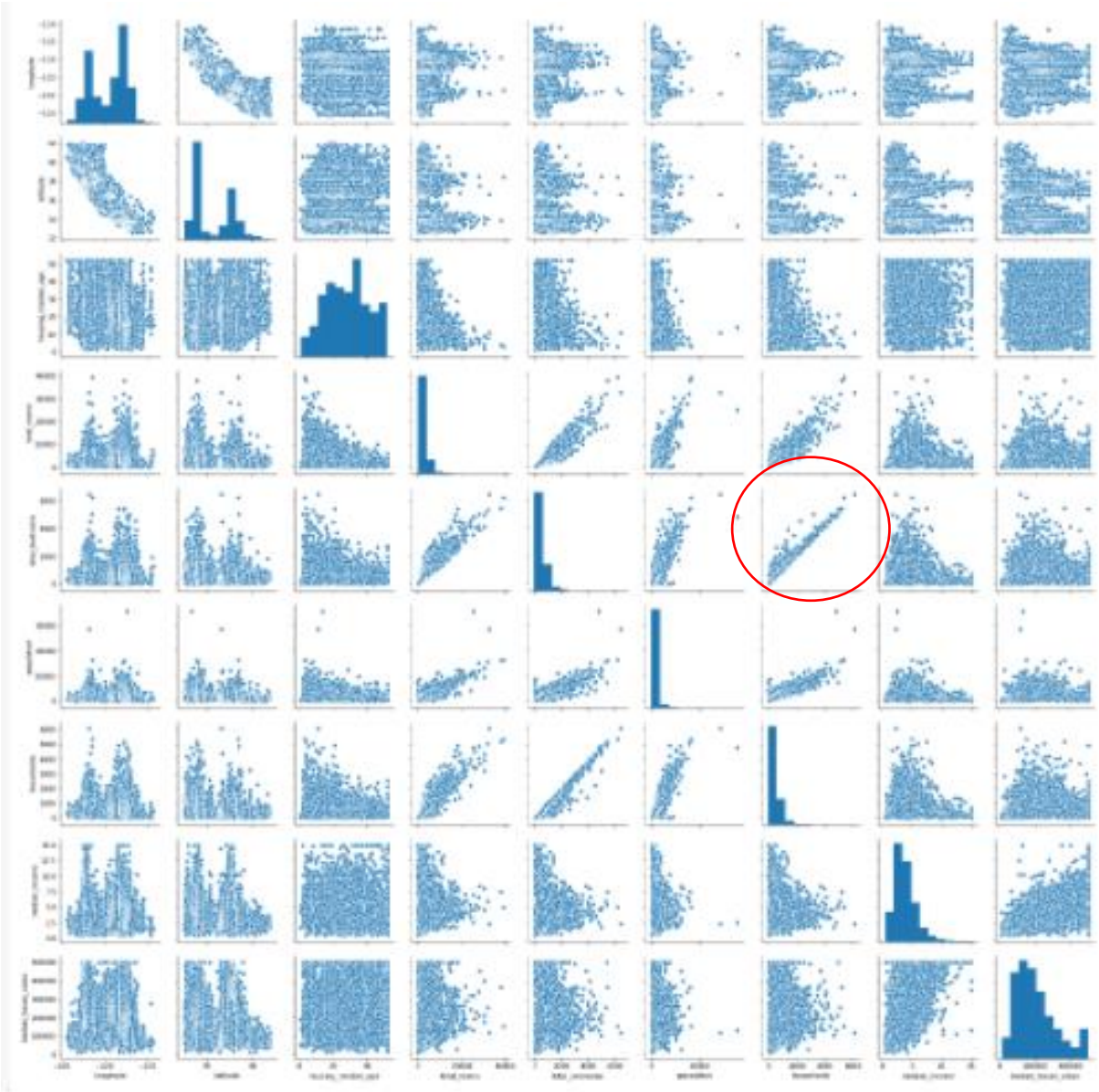
قسمت d) نمودار رسم شده مطابق زیر می باشد



سوال ۲)

قسمت a) همانطور که در نمودار های بدست آمده مشخص است در برخی از نمودار ها توزیع یک متغیر برحسب متغیر دیگر کاملا پراکنده می باشد این در حالی است که در برخی از نمودار های دیگر توزیع یک متغیر برحسب متغیر دیگر همانند خط و یا نزدیک به آن می باشد هرچه نمودار بدست آمده به یک خط یا منحنی شبیه تر بوده و شکل پراکنده نداشته باشد بدان معناست که دو متغیر مربوط به آن دارای correlation بیشتری می باشند

قسمت b) همانطور که از تصویر زیر مشخص است دو متغیر households , total_bedroom بیشترین میزان correlation را دارا هستند چرا که نمودار نظیر آن ها به شکل خط نزدیک تر بوده و کمترین میزان پراکندگی را نسبت به سایر نمودار ها داراست.



قسمت c) در روش pearson دو اصطلاح وجود دارد یکی ضریب همبستگی و دیگریP_value می باشد.

ضریب همبستگی یا همان correlation coefficient یک عدد بین -۱ تا ۱ است که معین می سازد دو متغیر از مجموعه داده ها تا چه میزان به هم مربوط هستند. هرچه correlation coefficient مربوط به دو متغیر به عدد ۱ نزدیک تر باشد بدان معناست که با افزایش مقدار یک متغیر متغیر دیگر نیز به صورت مرتبط مقدار آن افزایش می یابد و ضریب همبستگی هر چه به عدد -۱ نزدیک تر باشد بدان معناست که دو متغیر رابطه عکس داشته و با افزایش مقدار یکی مقدار دیگری به نحوی مرتبط کاهش می یابد هرچه ضریب همبستگی به عدد ۰ نزدیک تر باشد به این معناست که دو متغیر فاقد رابطه خاصی هستند.

مقدار p_value نیز به این صورت قابل تعریف است : اگر در مسئله کشف رابطه میان دو متغیر یک فرضیه صفر مبنی بر اینکه دو متغیر مورد نظر هیچ رابطه ای ندارند در نظر بگیریم آنگاه p_value مقداری بین ۰ و ۱ است که نمایانگر احتمال وجود داده ها برمبنای درست بودن فرضیه صفر است. یعنی اگر مقدار p_value کوچک باشد به این معناست که احتمال درست بودن فرضیه صفر پایین بوده و لذا متغیر ها دارای ارتباط هستند و درصورتی که بزرگ باشد به این معناست که احتمال درست بودن فرضیه صفر زیاد بوده و لذا متغیر ها فاقد ارتباط هستند.

http://www.eecs.qmul.ac.uk/~norman/blog_articles/p_values.pdf

قسمت d) با استفاده از تابع pearsonr(x, y) کتابخانه scipy.stats می تواند میزان همبستگی میان دو متغیر را بدست آورد به علاوه به کمک تابع corr و استفاده از متد pearsonr در آن می توان به نتیجه مشابه دست یافت.

```
0.6880752079585478
```

قسمت e) با استفاده از تابع spearmanr (x, y) کتابخانه scipy.stats می تواند میزان همبستگی میان دو متغیر را بدست آورد به علاوه به کمک تابع corr و استفاده از متد spearman در آن می توان به نتیجه مشابه دست یافت.

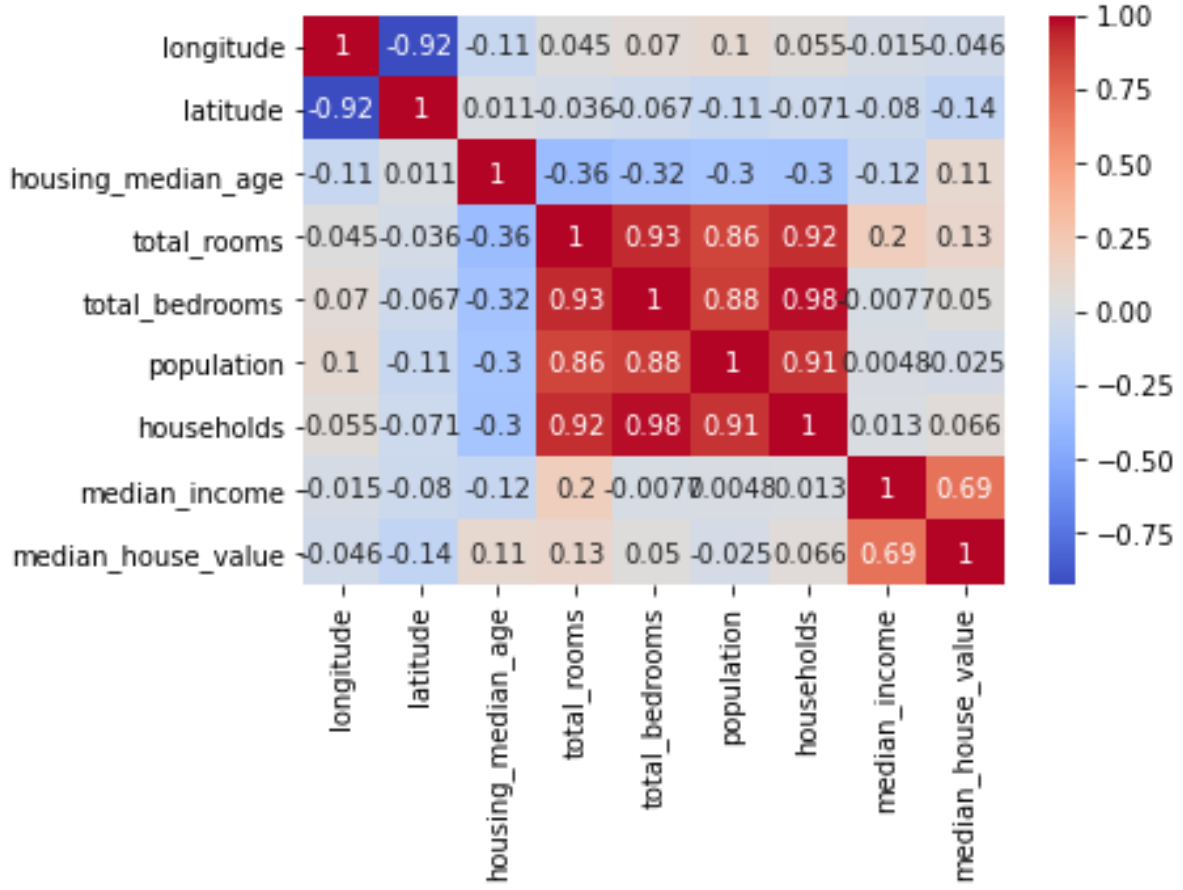
```
- 0.3571622692099669
```

قسمت f) تابع corr به عنوان پارامتر ورودی متدهای {'pearson', 'kendall', 'spearman'} را دریافت می کند. در صورتی که بدون تعیین متد فراخوانی شود به صورت پیش فرض از متد pearson استفاده می کند. به همین سبب مقدار بدست آمده برای همبستگی در این قسمت با مقدار بدست آمده در قسمت d یکسان می باشد.

```
0.688075207958548
```

قسمت g) تابع heatmap رسم شده به صورت زیر خواهد بود.

همانطور که مشخص است نزدیک ترین عدد به ۱ یا -۱ در داخل نمودار عدد 0.98 می باشد که همبستگی میان دو متغیر total_bedroom, households را نمایش می دهد پس نتیجه بدست آمده در قسمت b تایید می گردد.



سوال ۳)

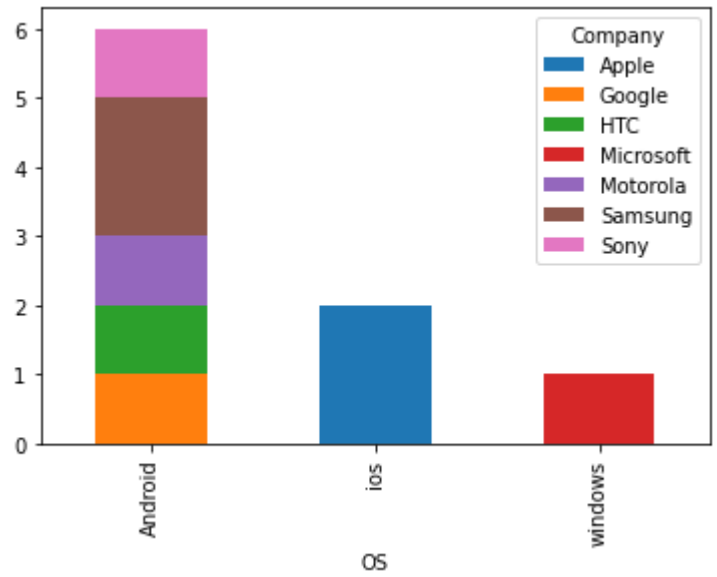
قسمت a) جدول همسانی دو متغیر را با استفاده از متد crosstab کتابخانه pandas رسم کرده و این جدول مطابق زیرخواهد بود.

Capacity	16	32	64	128
Company				
Apple	0	1	0	1
Google	0	0	0	1
HTC	0	0	1	0
Microsoft	0	1	0	0
Motorola	1	0	0	0
Samsung	1	0	1	0
Sony	1	0	0	0

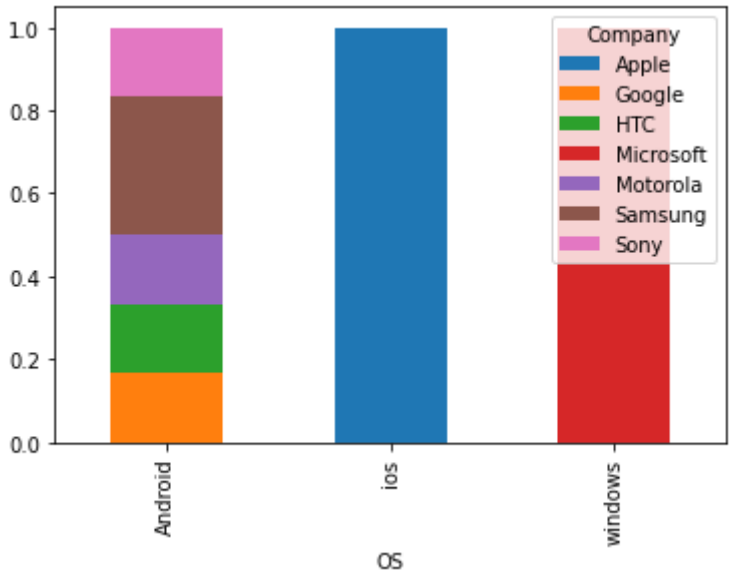
قسمت b) جدول همسانی سه متغیر را نیز با استفاده از متد crosstab کتابخانه pandas رسم کرده و این جدول مطابق زیرخواهد بود.

	Company	Apple	Google	HTC	Microsoft	Motorola	Samsung	Sony
Weight	inch							
112.0	4.0	1	0	0	0	0	0	0
138.0	4.7	1	0	0	0	0	0	0
143.0	5.0	0	1	0	0	0	0	0
144.5	5.0	0	0	0	0	1	0	0
145.0	5.1	0	0	0	0	0	1	0
149.0	5.8	0	0	0	0	0	1	0
150.0	5.2	0	0	0	1	0	0	0
170.0	5.7	0	0	1	0	0	0	0
180.0	5.5	0	0	0	0	0	0	1

قسمت c) جدول همسانی دو متغیر را با استفاده از متد crosstab کتابخانه pandas بدست آورده و نمودار نظیر این جدول را به استفاده از کتابخانه matplotlib رسم می کنیم نمودار بدست آمده مطابق زیر خواهد بود.



می توانیم عمل یکسان سازی طول نمودار ها را نیز انجام داده و مقایسه بهتری داشته باشیم



سوال ۴)

قسمت a) با استفاده از متد train_test_split در کتابخانه sklearn.model_selection داده ها را به دو دسته تست و یادگیری تقسیم می کنیم به طوری که داده های تست ۲۰ درصد از کل داده ها باشند.

```
thyroid_trian , thyroid_test = train_test_split(df, test_size = 0.2, random_state = 7)
```

قسمت b) استفاده از پارامتر stratify موجب می گردد توزیع متغیر هدف در داده های test, train حاصل از تابع train_test_split متناسب با توزیع این متغیر در مجموعه داده های اولیه باشد.

قسمت c) خیر اگر توزیع متغیر Outcome را در داده های test, train بدست آوریم آنگاه مشاهده خواهیم کرد که در داده های test توزیع مقدار 1.0 برای متغیر Outcome حدود 78 درصد و توزیع مقدار 2.0 برای این متغیر حدود 21 درصد است در حالی که در داده

های train توزیع مقدار 1.0 برای متغیر Outcome حدود 84 درصد و توزیع مقدار 2.0 برای همین متغیر حدود 16 درصد است. به این ترتیب داده های تست و یادگیری توزیع یکسانی از متغیر هدف را دارا نیستند.

قسمت d)

Resampling The Training Set

دو رویکرد عمده برای resample کردن داده های imbalanced موجود می باشد.

۱- Oversampling : به صورت رندوم رکورد ها و موجودیت هایی که مقدار متغیر هدف نظیر آن ها در دسته اقلیت قرار میگیرد را در دیتاست تکرار کنیم. این روش برای دیتاست های کوچک مناسب می باشد.

۲- Under sampling : به صورت رندوم رکورد ها و موجودیت هایی که مقدار متغیر هدف نظیر آن ها در دسته اکثریت قرار میگیرد را حذف کنیم. این روش برای دیتاست های بزرگ مناسب می باشد.

Cluster the abundant class

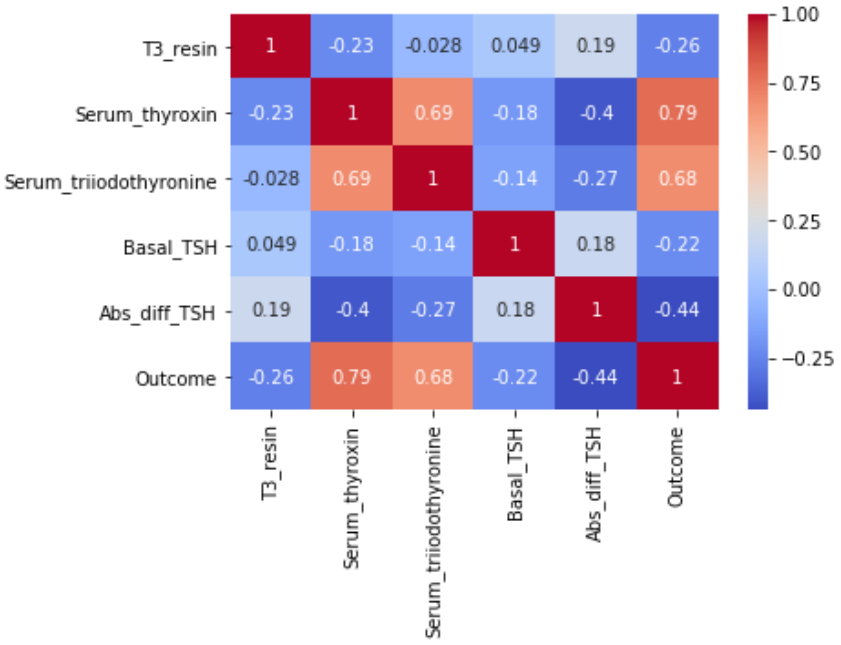
رویکردی است که در آن به جای استفاده از نمونه های رندوم برای پوشش دادن کلیه نمونه های یادگیری ابتدا نمونه های موجود را در r گروه دسته بندی کرده و در هر دسته نیز r نمونه قرار می دهیم سپس تنها نمونه مرکزی هر دسته نگه داشته شده و با دسته نادر و نمونه های مرکزی مدل آموزش داده می شود.

https://www.kdnuggets.com/2017/06/7-techniques-handle-imbalanced-data.html

سوال ۵)

قسمت a) ابتدا تمام مقادیری که به عنوان مقدار Null قابل قبول هستند را به NaN تبدیل کرده (به دلیل وجود مقداری چون ? در داخل دیتاست) و سپس رکورد های حاوی مقدار null برای ستون Outcome را حذف می کنیم چراکه Outcome متغیر هدف بوده و نمی توانیم مقادیر null این ستون را با میانگین داده های دیگر جایگذاری کنیم. در گام بعدی ۳ ستونی که نوع آن ها از نوع object هست را به float تبدیل کرده و سپس به کمک متد SimpleImputer داده های null را با میانگین ستون جایگذاری می کنیم.

قسمت b) نمودار heatmap رسم شده به صورت زیر می باشد همانطور که مشخص است دو متغیر serum_thyroxin و Outcome با مقدار 0.79 بیشترین میزان همبستگی را دارا می باشند.



قسمت c) بله دیتاست مورد نظر imbalanced می باشد چرا که اگر به کمک متد value_counts() توزیع متغیر outcome را بررسی کنیم خواهیم دید که توزیع دو مقدار 1.0 و 2.0 یکسان نمی باشد.

```
In [237]:
# 5_c

MyDataFram['Outcome'].value_counts()
# Dataset is impbalance
```

```
Out[237]: 1.0    144
          2.0     30
          Name: Outcome, dtype: int64
```

لذا با استفاده از resample کردن داده های با مقدار outcome برابر 2.0 توزیع داده ها را یکسان می سازیم به نحوی که ۵۰ درصد داده ها دارای مقدار outcome برابر 1.0 و 50 درصد دیگر دارای مقدار 2.0 باشند.

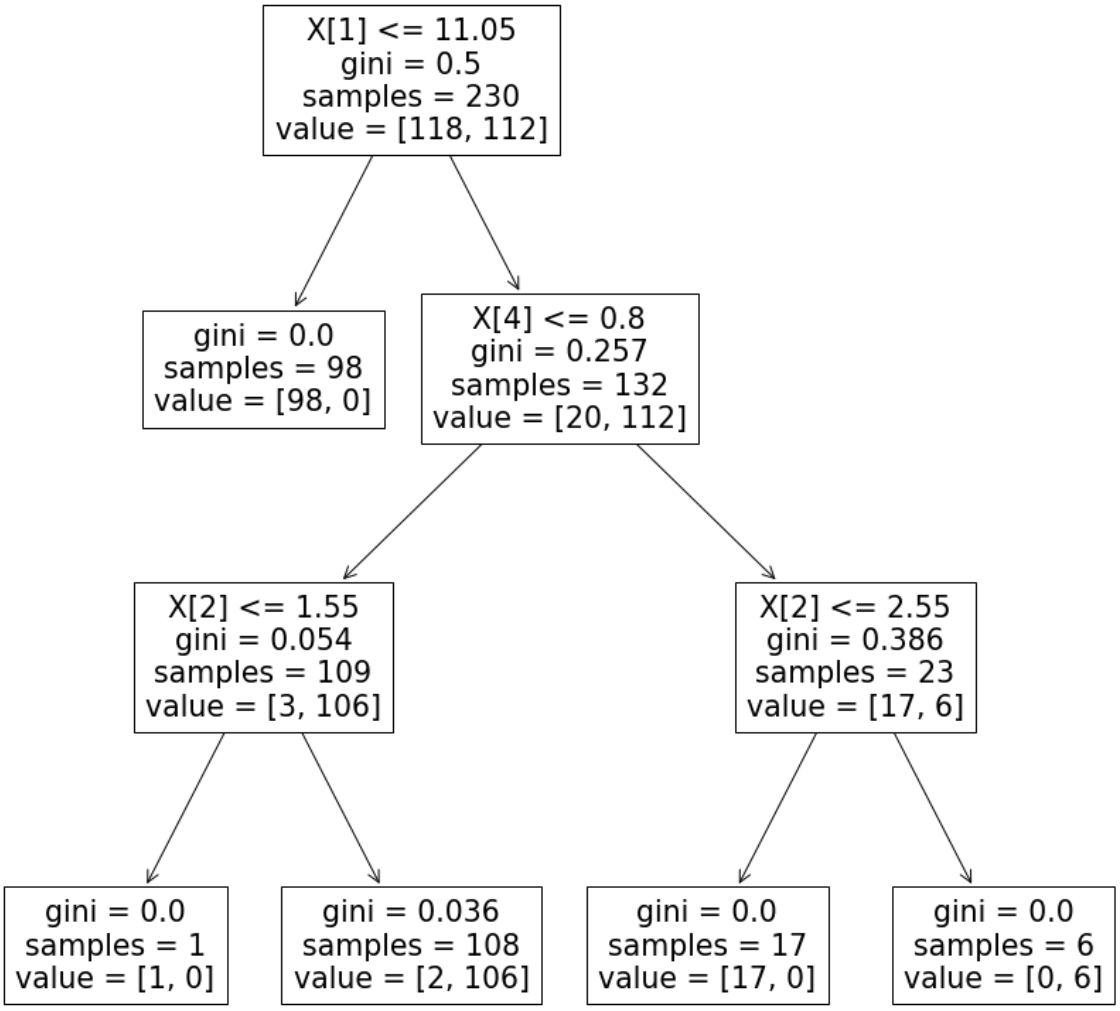
```
In [238]:
To_Resample = MyDataFram[MyDataFram['Outcome'] == 2]
After_Resample =To_Resample.sample(n = 114, replace = True)
MyDataFram_New = pd.concat([After_Resample, MyDataFram])
```

```
In [239]:
MyDataFram_New['Outcome'].value_counts()
# Dataset is balance now
```

```
Out[239]: 1.0    144
          2.0    144
          Name: Outcome, dtype: int64
```

قسمت d) با استفاده از متد train_test_split در کتابخانه sklearn.model_selection داده ها را به دو دسته تست و یادگیری تقسیم می کنیم به طوری که داده های تست ۲۰ درصد از کل داده ها باشند.

قسمت e) با استفاده از متد DecisionTreeClassifier در کتابخانه sklearn.tree درخت تصمیم را برای داده های یادگیری تشکیل خواهیم داد. درخت بدست آمده مشابه زیر خواهد بود :



قسمت f) به کمک متد score میزان دقت را برای داده های تست داده شده به مدل درخت سنجش می کنیم. این دقت در هر بار تقسیم داده ها به دو دسته test, train و تشکیل درخت تصمیم و نهایتا سنجش داده های تست به کمک مدل بدست آمده متفاوت بوده و عددی بین 0.9 تا 1 می باشد بدان معنا که دقت مدل هربار در حدود ۹۰ الی ۱۰۰ درصد است.

قسمت g) بنابر آزمایش های متعدد تا یک آستانه ای هر چه max_depth درخت عدد بزرگتری باشد دقت داده های تست بهتر شده و به عدد ۱ نزدیک تر می گردد. چرا که تعداد دفعات دسته بندی افزایش یافته و با دقت بیشتری دسته بندی می تواند صورت پذیرد. برای مثال بیشترین میزان دقت بدست آمده طی آزمایش زیر دقت 1.0 است که با max_depth = 2 بدست آمده است.

```
accuracy for max_depth =  1  :  0.9310344827586207
accuracy for max_depth =  2  :  1.0
accuracy for max_depth =  3  :  1.0
accuracy for max_depth =  4  :  1.0
accuracy for max_depth =  5  :  1.0
accuracy for max_depth =  6  :  1.0
accuracy for max_depth =  7  :  1.0
accuracy for max_depth =  8  :  1.0
accuracy for max_depth =  9  :  1.0


Best Max_depth =  2
```

قسمت h) متد feature_importance تعیین کننده میزان اهمیت هر feature می باشد هر چه عدد عدد مربوطه بزرگتر باشد بدان معناست که در درخت تصمیم feature مربوطه در سطح بالاتری به عنوان جداکننده قرار میگیرد. برای مثال در درخت تصمیم رسم شده در قسمت e اگر این متد را فراخوانی کنیم اعداد زیر بدست خواهد آمد :

```
array([0, 0.7295982 , 0.09710728, 0, 0.17329452])
```

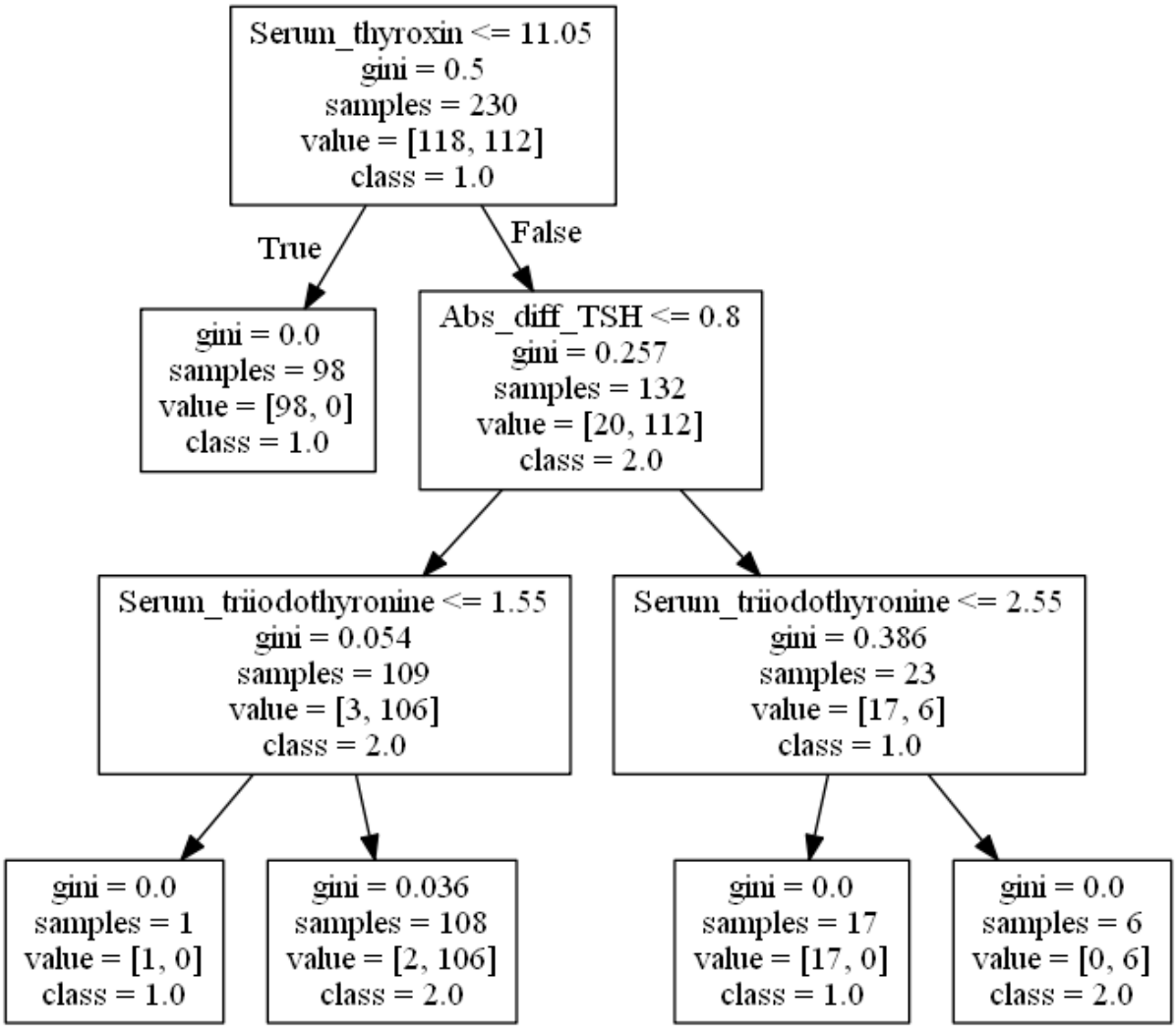
همانطور که مشخص است بیشترین مقدار مربوط به x1 است که در ریشه درخت تصمیم به عنوان جداکننده ظاهر شده است. عدد دوم مربوط به x4 است که در سطح دوم درخت ظاهر شده و عدد سوم مربوط به x2 است که در سطح سوم درخت ظاهر شده است.

قسمت i) از متد های DecisionTreeClassifier و export_graphviz استفاده کرده و ابتدا درخت تصمیم را برمبنای داده های یادگیری تشکیل داده و سپس این درخت را در فایل Decision_Tree.dot ذخیره می کنیم.

```
# 5_i

Decision_Tree = DecisionTreeClassifier(criterion = "gini", max_depth = best_max_depth).fit(x_train, y_train)
export_graphviz(Decision_Tree, out_file = "Decision_Tree.dot", class_names = y_names)
```

قسمت j) از متد graph_from_dot_file کتابخانه pydotpluse استفاده کرده و یک نمونه از گراف بدست آمده مشابه زیر خواهد بود


Serum_thyroxin <= 11.05
gini = 0.5
samples = 230
value = [118, 112]
class = 1.0
True
False
gini = 0.0
samples = 98
value = [98, 0]
class = 1.0
Abs_diff_TSH <= 0.8
gini = 0.257
samples = 132
value = [20, 112]
class = 2.0
Serum_triiodothyronine <= 1.55
gini = 0.054
samples = 109
value = [3, 106]
class = 2.0
Serum_triiodothyronine <= 2.55
gini = 0.386
samples = 23
value = [17, 6]
class = 1.0
gini = 0.0
samples = 1
value = [1, 0]
class = 1.0
gini = 0.036
samples = 108
value = [2, 106]
class = 2.0
gini = 0.0
samples = 17
value = [17, 0]
class = 1.0
gini = 0.0
samples = 6
value = [0, 6]
class = 2.0