به نام خدا

پروژه دوم درس هوش مصنوعی

سوال اول

محمد امین سلطانی چم حیدری

۸۱۰۶۰۱۰۸۱

استاد مربوطه:

دکتر مسعود شریعت پناهی

بهار۱۴۰۲

بخش اول: پیش‌بینی نوع تومور سرطان سینه

در سال‌های اخیر استفاده از الگوریتم‌های یادگیری ماشین نقش مهمی در افزایش سرعت و دقت تشخیص نوع و شدت بیماری‌ها داشته‌اند. دادگان (dataset)ی که در این بخش مورد استفاده قرار می‌گیرد (فایل breast_cancer.csv) شامل اطلاعات ۶۸۴ بیمار با تشخیص سرطان سینه است. هر نمونه شامل ۹ ویژگی (مانند مشخصات تومور و کیفیت بافت اطراف آن) (ستون‌های ۱ تا ۹) و یک خروجی (نوع تومور) (ستون دهم) است که در این ستون عدد ۲ به معنای تومور خوش‌خیم و عدد ۴ به معنای تومور بدخیم است.

در این سوال برای نوشتن کد از jupyter notebook استفاده شده است. همچنین با توجه به نکته ذکر شده در پانوشت سوال از توضیح جزئیات کد خودداری شده است.

برای حل موارد الف تا ه ابتدا کتابخانه های مورد نیاز در پایتون فراخوانی می شوند.

required libraries

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import pylab as pl
from sklearn import preprocessing
```

matplotlib ← نمایش data و رسم نمودار

pandas ← کار با داده های ساختار یافته مانند جداول،فایل csv و excel و...

Numpy ← انجام عملیات ریاضی،آماری و عددی

Pylab ← تجسم داده های علمی و عملی.ترکیبی از matplotlib و numpy می باشد

Sickit_learn ← یادگیری ماشین و تحلیل داده

در قدم بعدی فایل excel مورد نیاز شامل اطلاعات بیماران فراخوانی می شود.

open file

```
In [35]: Breast_cancer_df = pd.read_csv("breast_cancer.csv")
         Breast_cancer_df.head()
```

Out[35]:

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
0	5	1	1	1	2	1	3	1	1	2
1	5	4	4	5	7	10	3	2	1	2
2	3	1	1	1	2	2	3	1	1	2
3	6	8	8	1	3	4	3	7	1	2
4	4	1	1	3	2	1	3	1	1	2

در مرحله بعد برای اثبات عدم وجود نقصی و پرتی در داده ها اطلاعات مربوط به آن ها فراخوانی می شود.از هر دو دستور زیر جداگانه می توان به این نتیجه رسید که در داده ها نقص و پرتی وجود ندارد.

در دستور اول با توجه به اینکه همه داده ها non-null و از نوع int64 هستند می توان نتیجخه گرفت که این دسته از داده قبلا پاک سازی شده اند و مشکلی ندارند.

در دستور دوم نیز باتوجه به اینکه در همه ستون های جدول اعداد وجود دارند می توان این نتیجه را گرفت.

clear data

```
In [41]: print (Breast_cancer_df.info())
         Breast_cancer_df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 683 entries, 0 to 682
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Clump Thickness              683 non-null    int64
 1   Uniformity of Cell Size      683 non-null    int64
 2   Uniformity of Cell Shape     683 non-null    int64
 3   Marginal Adhesion            683 non-null    int64
 4   Single Epithelial Cell Size  683 non-null    int64
 5   Bare Nuclei                  683 non-null    int64
 6   Bland Chromatin              683 non-null    int64
 7   Normal Nucleoli              683 non-null    int64
 8   Mitoses                      683 non-null    int64
 9   Class                        683 non-null    int64
dtypes: int64(10)
memory usage: 53.5 KB
None
```

Out[41]:

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
count	683.000000	683.000000	683.000000	683.000000	683.000000	683.000000	683.000000	683.000000	683.000000	683.000000
mean	4.442167	3.150805	3.215227	2.830161	3.234261	3.544656	3.445095	2.869693	1.603221	2.699854
std	2.820761	3.065145	2.988581	2.864562	2.223085	3.643857	2.449697	3.052666	1.732674	0.954592
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	2.000000
25%	2.000000	1.000000	1.000000	1.000000	2.000000	1.000000	2.000000	1.000000	1.000000	2.000000
50%	4.000000	1.000000	1.000000	1.000000	2.000000	1.000000	3.000000	1.000000	1.000000	2.000000
75%	6.000000	5.000000	5.000000	4.000000	4.000000	6.000000	5.000000	4.000000	1.000000	4.000000
max	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	4.000000

در این قدم نام ستون های جدول استخراج می شود تا در مراحل بعد از آن متغیر ها و خروجی مسئله به سادگی تعریف شود.

```
Breast_cancer_df.columns
```

```
Index(['Clump Thickness', 'Uniformity of Cell Size',
       'Uniformity of Cell Shape', 'Marginal Adhesion',
       'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin',
       'Normal Nucleoli', 'Mitoses', 'Class'],
      dtype='object')
```

با توجه به اینکه کتابخانه pandas آرایه ها را به صورت str فراخوانی می کند و سوال نیاز به داده های ریاضی دارد در این قسمت برای تعریف آرایه های ورودی و خروجی در متغیرهای , x و y از دستور numpy مطابق شکل زیر استفاده می شود.

defin x & y

```
In [21]: X = np.asarray(Breast_cancer_df[['Clump Thickness','Uniformity of Cell Size','Uniformity of Cell Shape','Marginal Adhesion','
         X[0:10]
```

```
Out[21]: array([[ 5,  1,  1,  1,  2,  1,  3,  1,  1],
                [ 5,  4,  4,  5,  7, 10,  3,  2,  1],
                [ 3,  1,  1,  1,  2,  2,  3,  1,  1],
                [ 6,  8,  8,  1,  3,  4,  3,  7,  1],
                [ 4,  1,  1,  3,  2,  1,  3,  1,  1],
                [ 8, 10, 10,  8,  7, 10,  9,  7,  1],
                [ 1,  1,  1,  1,  2, 10,  3,  1,  1],
                [ 2,  1,  2,  1,  2,  1,  3,  1,  1],
                [ 2,  1,  1,  1,  2,  1,  1,  1,  5],
                [ 4,  2,  1,  1,  2,  1,  2,  1,  1]], dtype=int64)
```

```
In [22]: y = np.asarray(Breast_cancer_df['Class'])
         y [0:10]
```

```
Out[22]: array([2, 2, 2, 2, 2, 4, 2, 2, 2, 2], dtype=int64)
```

با توجه به خواسته ی قسمت الف داده های ستون class که شامل اعداد ۲ و ۴ بودند و در متغیر y تعریف شدند، مطابق دستور زیر به آرایه های صفر و یکی تبدیل می شوند.

changing y array to 0 & 1

```
In [23]: y = np.where(y == 4, 1, 0)
         y[0:10]
```

```
Out[23]: array([0, 0, 0, 0, 0, 1, 0, 0, 0, 0])
```

همچنین خواسته ی دیگر قسمت الف normalize کردن یا عادی سازی داده های ورودی یا همان آرایه های متغیر x می باشد که مطابق دستور زیر داده های متغیر x عادی سازی یا همان normalize شده اند.

normalizing x

In [24]:
```
print(X[0:10])
Scaler= preprocessing.StandardScaler().fit(X)
X =Scaler.transform(X.astype(float))
print(X[0:10])
```

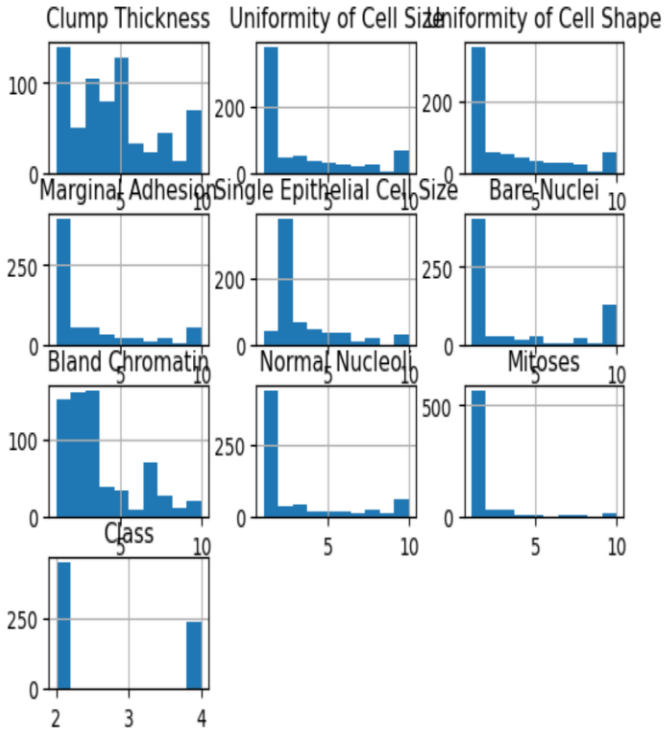
```
[[ 5  1  1  1  2  1  3  1  1]
 [ 5  4  4  5  7 10  3  2  1]
 [ 3  1  1  1  2  2  3  1  1]
 [ 6  8  8  1  3  4  3  7  1]
 [ 4  1  1  3  2  1  3  1  1]
 [ 8 10 10  8  7 10  9  7  1]
 [ 1  1  1  1  2 10  3  1  1]
 [ 2  1  2  1  2  1  3  1  1]
 [ 2  1  1  1  2  1  1  1  5]
 [ 4  2  1  1  2  1  2  1  1]]
[[ 0.19790469 -0.70221201 -0.74177362 -0.63936553 -0.5556085  -0.69885309
  -0.18182716 -0.61292736 -0.34839971]
 [ 0.19790469  0.27725185  0.26278299  0.75803177  1.69516613  1.77286724
  -0.18182716 -0.28510482 -0.34839971]
 [-0.51164337 -0.70221201 -0.74177362 -0.63936553 -0.5556085  -0.4242175
  -0.18182716 -0.61292736 -0.34839971]
 [ 0.55267873  1.58320366  1.6021918  -0.63936553 -0.10545357  0.12505369
  -0.18182716  1.3540079  -0.34839971]
 [-0.15686934 -0.70221201 -0.74177362  0.05933312 -0.5556085  -0.69885309
  -0.18182716 -0.61292736 -0.34839971]
 [ 1.26222679  2.23617957  2.2718962   1.80607975  1.69516613  1.77286724
   2.26925078  1.3540079  -0.34839971]
 [-1.22119144 -0.70221201 -0.74177362 -0.63936553 -0.5556085   1.77286724
  -0.18182716 -0.61292736 -0.34839971]
 [-0.8664174  -0.70221201 -0.40692142 -0.63936553 -0.5556085  -0.69885309
  -0.18182716 -0.61292736 -0.34839971]
 [-0.8664174  -0.70221201 -0.74177362 -0.63936553 -0.5556085  -0.69885309
  -0.99885314 -0.61292736  1.96186243]
 [-0.15686934 -0.37572406 -0.74177362 -0.63936553 -0.5556085  -0.69885309
  -0.59034015 -0.61292736 -0.34839971]]
```

ب) توزیع دادگان را برای تمام ۹ ویژگی بدست آورید. داده‌های با نوع تومور مختلف را با رنگ های متفاوت از یکدیگر نمایش دهید. پراکندگی داده‌ها را بر حسب ویژگی‌های مختلف نشان دهید. بر پایهٔ این پراکندگی‌ها برداشت خود را از ماهیت داده‌ها بیان کنید (راهنمایی: برای نمایش توزیع داده‌ها میتوانید از نمودارهای نقشه گرمایی، گسسته، هیستوگرام یا جعبه ای استفاده کنید.)

برای بدست آوردن توزیع دادگان برای هر ویژگی ابتدا تابعی شامل ویژگی های مد نظر تعریف کرده و سپس با استفاده از دستور رسم هیستوگرام برای هر ویژگی تابع هیستوگرام رسم می شود.

Obtaining the data distribution

In [96]:
```
Bcf = Breast_cancer_df[['Clump Thickness', 'Uniformity of Cell Size','Uniformity of Cell Shape', 'Marginal Adhesion', 'Single
Bcf.head(3)
viz = Bcf[['Clump Thickness', 'Uniformity of Cell Size','Uniformity of Cell Shape', 'Marginal Adhesion', 'Single Epithelial C
viz.hist()
plt.show()
```



با توجه به نمودارهای بدست آمده می توان برای هر ویژگی پراکندگی داده ها را از روی شکل بالا مشاهده کرد. مثلا برای ویژگی Clump Thickness پراکندگی داده ها بین ۱ تا ۱۰ به طور نسبتا یکنواخت تری از ویژگی Bare Nuclei می باشد. می توان از این موضوع نتیجه گرفت که ویژگی Clump Thickness کمتر از Bare Nuclei مدل ما را دچار خطا میکند.

البته داده های مشاهده شد پس از normalize کردن نشان داده شده اند.اگر قبل از آن نمودار ها را رسم می کردیم پراکندگی بیشتر بود و همچنین دامنه تغییرات ویژگی ها نیز با یکدیگر متفاوت می بود.

ج) رابطه و تاثیرگذاری هر کدام از پارامترها بر نوع تومور را پیدا کنید. (راهنمایی: می‌توانید از معیارهای آماری مانند کوریلیشن استفاده کنید تا میزان تاثیر داده‌ها بر روی خروجی و حتی رابطهٔ آن‌ها با یکدیگر را مشاهده کنید. برای نمایش هم می‌توانید از pair-plot در کتابخانه seaborn استفاده کنید.)

برای محاسبه تاثیر هرپارامتر بر نوع تومور مقدار کورلیشن هر پارامتر بدست آورده شده است.

ضریب (correlation coefficient) یا ضریب همبستگی نشان دهنده قدرت و جهت رابطه بین دو متغیر می باشد و مقدار آن بین 1 و1- است. مقدار یک نشان دهنده رابطه مستقیم کامل بین دو متغیز است.یعنی با افزایش یک متغیر متغیر دیگر نیز افزایش می یابد و مقدار 1- نشان دهنده رابطه معکوس بین دو متغیر است.یعنی با افزایشیک متغیر،متغیر دیگر کاهش می یابد.مقدار صفر نشان دهنده عدم وجود هرگونه رابطه بین دو متغیر می باشد.

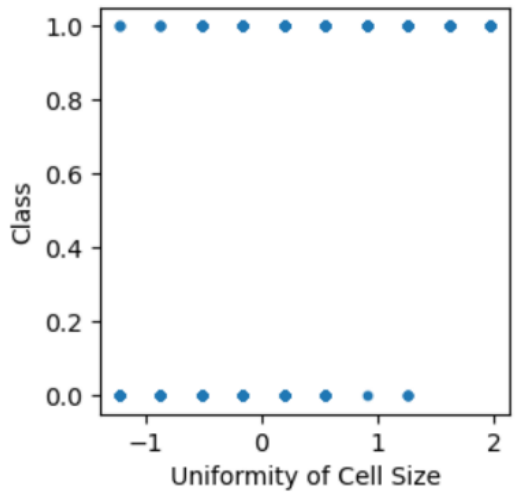
برای محاسبه ضریب همبستگی و نمایش نمودار تومور بر حسب هر متغیر مطابق کد زیر عمل می شود.

The effect of each parameter on the type of tumor

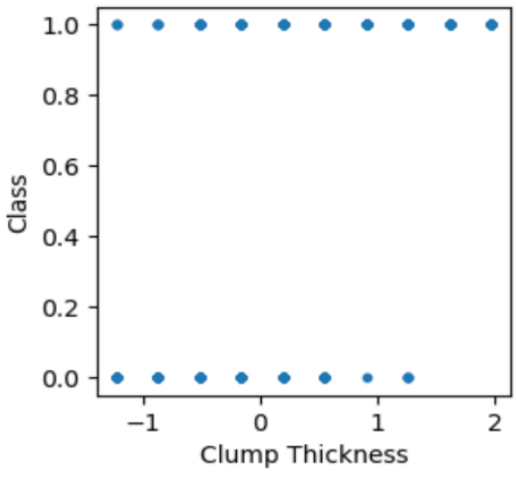
```
corr_coef = np.corrcoef(X[:, 0], y)[0, 1]
print("Correlation coefficient: ", corr_coef)
plt.figure(figsize=(3, 3), dpi=100)
plt.scatter(X[:, 0], y, s=10)
plt.xlabel("Clump Thickness")
plt.ylabel("Class")
plt.show()
```

در ادامه نمودار تاثیرگذاری و مقدار correlation coefficient (که برای هر نمودار بالای آن نوشته شده است) برای هر متغیر آورده شده است.

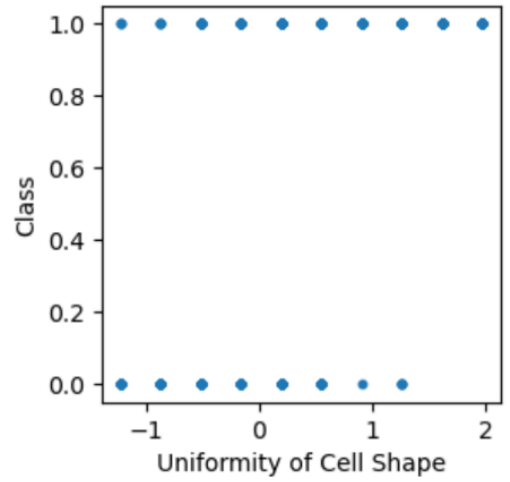
```
Correlation coefficient:  0.8208014428258776
```



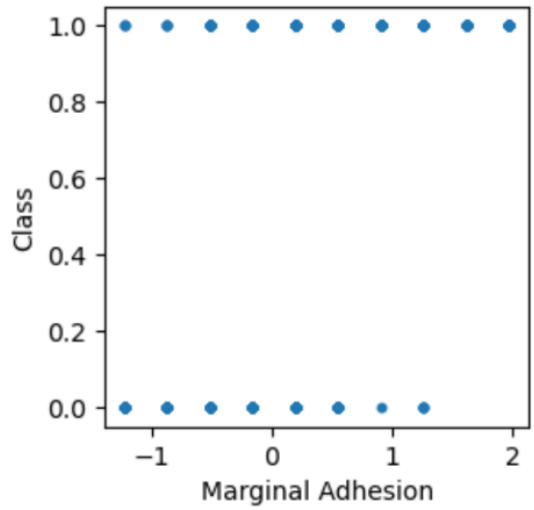

```
Correlation coefficient:  0.7147899263221612
```



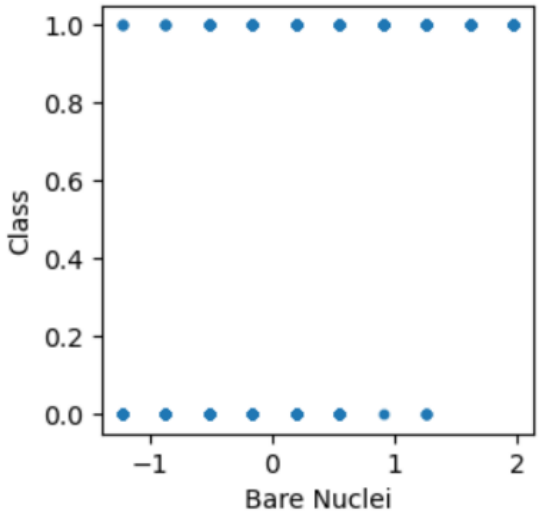
```
Correlation coefficient:  0.821890947688869
```



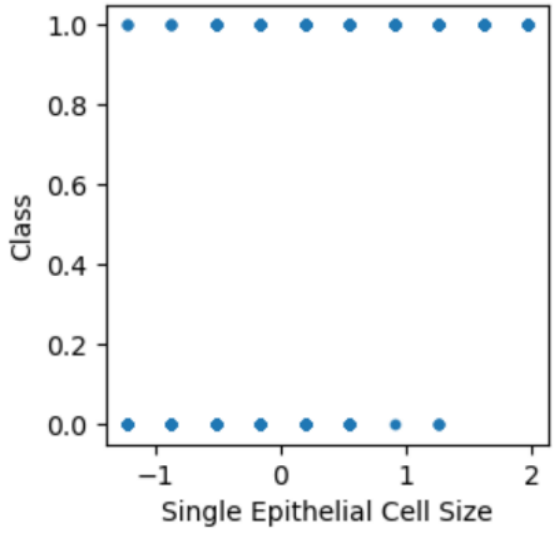

```
Correlation coefficient:  0.7062941354660857
```



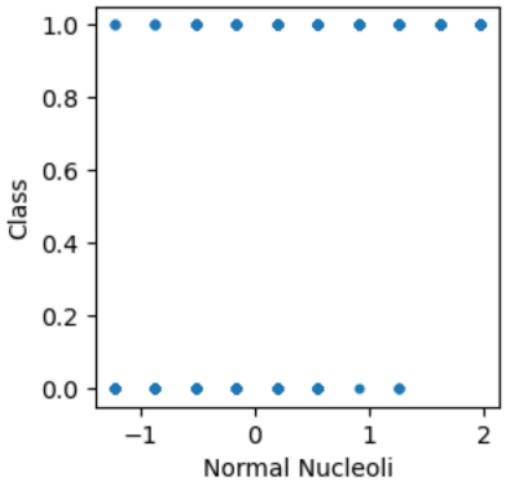

```
Correlation coefficient:  0.8226958729964602
```



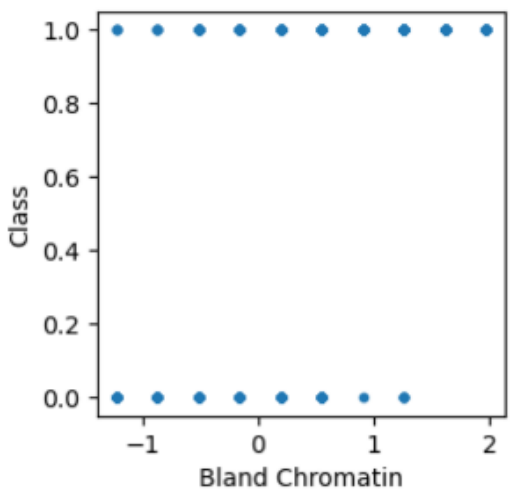

```
Correlation coefficient:  0.6909581590873196
```



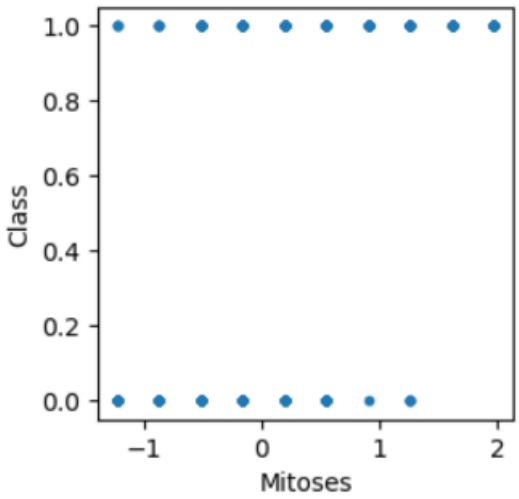
```
Correlation coefficient:  0.7186771878756363
```




```
Correlation coefficient:  0.7582275545334297
```




```
Correlation coefficient:  0.42344792129521286
```



همانگونه که مشخص است ضریب Bare Nuclei باتوجه به اینکه بیشترین correlation coefficient را دارد بیشترین تاثیر مستقیم را بر کلاس تومور می گذارد و ضریب Mitoses با کمترین correlation coefficient در بین متغیرها کمترین تاثیر را در کلاس تومور خواهد داشت.

نکته دیگر اینکه باتوجه به اینکه داده های خروجی بر حسب صفر و یک منظم شده و متغیر ها نیز normalize شدند لا نمودارها به خوبی قابل مقایسه هستند.

همچنین با توجه به اینکه تعداد data ما برای سوال بیشتر از نقاط مشخص شده در شکل بود می توان نتیجه گرفت نقطه ها روی هم افتاده اند و هر نقطه بیان گر چندنقطه ی روی هم می باشد.

د) داده ها را به سه بخش آموزش (training)، ارزیابی (validation) و آزمایش (test) تقسیم کنید. پیشنهاد می‌شود ۸۰٪ کل داده‌ها به آموزش، ۱۰٪ به ارزیابی و ۱۰٪ به آزمایش اختصاص داده شود. در گام بعد با استفاده از الگوریتم رگرسیون لجستیک، مدلی را برای پیش‌بینی خروجی تربیت کنید و سپس با استفاده از روش k-fold cross validation (با k=5) بهترین عملکرد مدل را بدست آورید (این روش در ادامهٔ درس معرفی خواهد شد.). (توضیح بیشتر: در یادگیری ماشین هر مدل برای تنظیم پارامترهای

Data separation

In [25]:
```
from sklearn.model_selection import train_test_split
X_train_valid, X_test, y_train_valid, y_test = train_test_split( X, y, test_size=0.1, random_state=20)
X_train, X_validation, y_train , y_validation = train_test_split(X_train_valid,y_train_valid, test_size=0.111,random_state=4
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
print ('validation set:', X_validation.shape, y_validation.shape)
```

```
Train set: (545, 9) (545,)
Test set: (69, 9) (69,)
validation set: (69, 9) (69,)
```

مطابق کد بالا و با استفاده از دستور Train-Test-split از کتابخانه scikit-learn داده های سوال شامل ورودی و خروجی ها به سه بخش تقسیم می شوند.مطابق خواسته سوال 80 درصد داده ها را برای تمرین و تعریف مدل، 10 درصد برای تست و ۱۰ درصد نیز برای ارزیابی در نظر گرفته می شود.

همچنین برای اینکه نتایج ذکر شده در فایل ارائه با نتایج فایل کد تفاوت نداشته باشد در اینجا حالت random state ثابت در نظر گرفته شده تا نتایج باهم همخوانی داشته باشند. در صورت تمایل برای تصادفی در نظر گرفته شدن داده ها در هر مرحله می توان حالت random state را پاک کرد.

```
In [26]:
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train,y_train)
LR
```

```
Out[26]:
LogisticRegression
LogisticRegression(C=0.01, solver='liblinear')
```

```
In [27]:
yhat = LR.predict(X_test)
yhat
```

```
Out[27]: array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1,
       0, 1, 1])
```

```
In [145]:
yhat_prob = LR.predict_proba(X_test)
print(yhat_prob[0:5])
```

```
[[0.03316452 0.96683548]
 [0.86437544 0.13562456]
 [0.8618102  0.1381898 ]
 [0.78344325 0.21655675]
 [0.91027743 0.08972257]]
```

مطابق خواسته صورت سوال با استفاده است رگرسیون لجستیک مدلی برای پیش بینی خروجی تعریف می شود.

در تابع y hat prob به محاسبه احتمال خروجی توسط رگرسیون لجستیکی پرداخته شده.اگر مقدار این تابع بزرگتر از نیم بود برابر یک و اگر کوچک تر از نیم بود برابر صفر قرار داده می شود.

در شکل بالا تابع y hat پیش بینی از حالت خروجی به صورت صفر و یک است. باتوجه به ماهیت رگرسیون لجستیکی که در دسته بنده classification قرار میگیرد و برای نتایج گسسته مورد استفاده قرار می گیرد در اینجا نیز خروجی سوال با توجه به مدل تربیت شده برابر صفر و یک قرار داده شده است.

```
from sklearn.model_selection import cross_val_score, KFold
from sklearn.linear_model import LogisticRegression

kfold = KFold(n_splits=10, shuffle=True, random_state=20)
model = LogisticRegression()
scores = cross_val_score(model, X, y, cv=kfold)
print('Accuracy:', scores.mean())
```

```
Accuracy: 0.9664109121909632
```

در پایان نیز صحت مدل تربیت شده با استفاده از روش k fold cross validation محاسبه شده که برابر 0.966 می باشد.

ه) بر روی داده‌های آزمایش، ماتریس سردرگمی را تشکیل دهید (این ماتریس نیز در ادامهٔ درس معرفی خواهد شد) و نتایج را تحلیل کنید. میزان دقت بدست آمده را مناسب می‌دانید یا خیر؟ پیشنهادهای خود را برای افزایش دقت ارائه دهید.

```
from sklearn.metrics import classification_report, confusion_matrix
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
print(confusion_matrix(y_test, yhat, labels=[1,0]))
```

```
[[21  1]
 [ 3 44]]
```

در این قسمت تابع سردرگمی تعریف شده و نتایج به صورت قابل مشاهده در آمده است.

در این ماتریس عدد 21 بیانگر TP می باشد.یعنی 21 مورد از داده های تست دارای تومورخوش خیم بوده اند و مدل آن را درست پیش بینی کرده است.

عدد 44 TN می باشد.یعنی 44 نفر تومورخیم داشته اند که مدل به درستی آن را پیش بینی کرده است

عدد 3 بیانگر FP می باشد.یعنی سه مورد را مدل دارای تومورخیم پیش بینی کرده در صورتی که تومورآنها خوش خیم بوده است.

عدد 1 نیز معرف FN می باشد.یعنی یک نفر دارای تومور بدخیم بوده است درحالی که مدل آن را خوش خیم پیش بینی کرده است.

در موارد حساس مانند این سوال اهمیت FN برای ما از همه بیشتر است زیرا درصورت که تومور بدخیم،خوش خیم پیش بینی شود خطر جانی دارد.لذا میزان دقت ما در این مورد نیاز به افزایش دارد.

برای این مورد میتوان داده های تست را افزایش داد.

همچنین می توان مدل های دیگر را برای پیش بینی امتحان کرد و بهترین مدل از لحاظ دقت را استفاده کرد.مثلا اگر از SVM برای تربیت این مدل استفاده شود در پایان می توان با به کار بردن Regularization دقت را افزایش داد.

و.....