Logistic Regression

> - ***Logistic regression is a classification algorithm used to predict a discrete set of classes.***
> - ***It is used to predict the outcomes of a categorical dependent variable***
> - ***Outcome should be discrete or categorical***

Example

- 0 or 1
- yes or no
- true or false
- high or low
- survied or not survied
- spam or not spam
- Low, Medium, High
- etc

Why logistic Regression

Suppose you have given data on "time spent on studying and exam scores by students". Linear Regression and logistic regression can predict different things-

Linear Regression could help us predict the student’s test score on a scale of 0 - 100.

Logistic Regression could help us to predict whether the student passed or failed.

Types of logistic regression

- Binary Logistic Regression (0/1) --> has only two 2 possible outcomes
- Multinomial Logistic Regression (Veg, Non-Veg, Vegan) --> Three or more categories without ordering
- Ordinal Logistic Regression (Low, Medium, High or movie rating 1 to 5) --> Three or more categories with ordering

Binary logistic regression

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
```

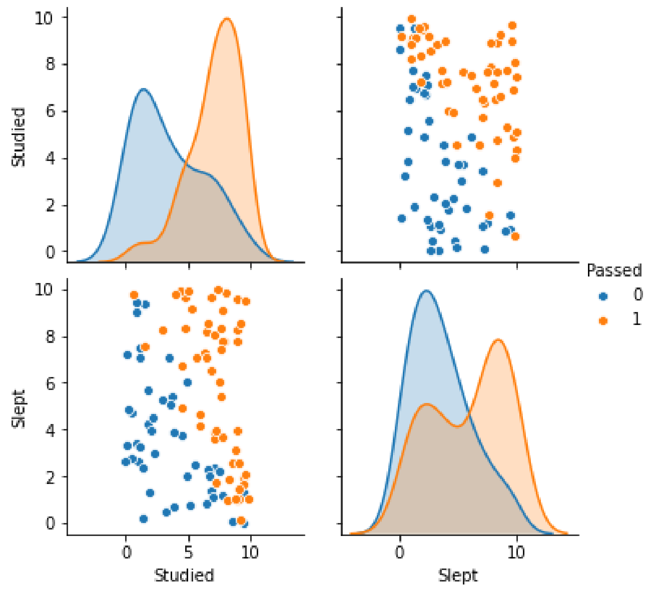
In [3]:
```
data = pd.read_csv('students_pperformance_classification.csv')
data.head()
```

Out[3]:

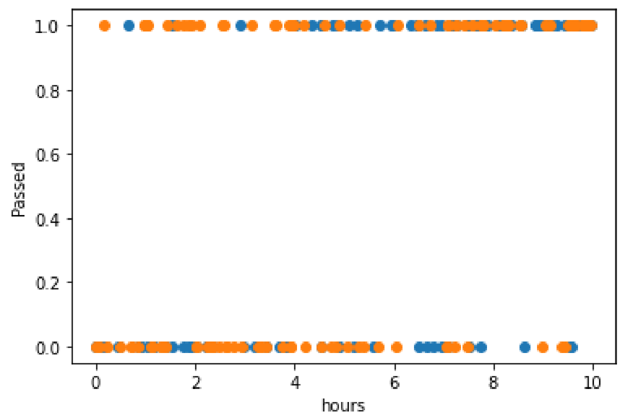
	Studied	Slept	Passed
0	4.855064	9.639962	1
1	8.625440	0.058927	0
2	3.828192	0.723199	0
3	7.150955	3.899420	1
4	6.477900	8.198181	1

In [4]:
```python
sns.pairplot(data,hue='Passed')
```

Out[4]: <seaborn.axisgrid.PairGrid at 0x17c3f110608>



In [5]:
```python
plt.scatter(data.Studied,data.Passed)
plt.scatter(data.Slept,data.Passed)
plt.xlabel('hours')
plt.ylabel('Passed')
plt.show()
```



Sigmoid Function

A solution for classification is logistic regression. Instead of fitting a straight line or hyperplane, the logistic regression model uses the logistic function to squeeze the output of a linear equation between 0 and 1.

The logistic function is defined as:

$$S(z) = \frac{1}{1 + e^{-z}}$$

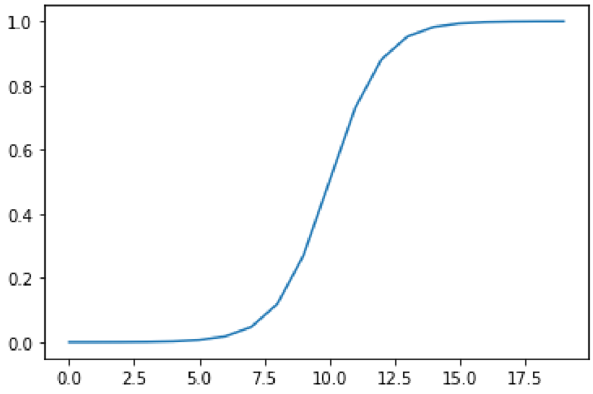
In [6]:
```python
def sigmoid(z):
    return 1.0 / (1 + np.exp(-z))
```

In [7]:
```
np.arange(-10,10,1)
```

Out[7]:
```
array([-10,  -9,  -8,  -7,  -6,  -5,  -4,  -3,  -2,  -1,   0,   1,   2,
         3,   4,   5,   6,   7,   8,   9])
```

In [8]:
```
x= list(map(sigmoid,np.arange(-10,10,1)))
plt.plot(x)
```

Out[8]:
```
[<matplotlib.lines.Line2D at 0x17c3f6ee608>]
```



In [9]:
```
x
```

Out[9]:
```
[4.5397868702434395e-05,
 0.00012339457598623172,
 0.0003353501304664781,
 0.0009110511944006454,
 0.0024726231566347743,
 0.0066928509242848554,
 0.01798620996209156,
 0.04742587317756678,
 0.11920292202211755,
 0.2689414213699951,
 0.5,
 0.7310585786300049,
 0.8807970779778823,
 0.9525741268224334,
 0.9820137900379085,
 0.9933071490757153,
 0.9975273768433653,
 0.9990889488055994,
 0.9996646498695336,
 0.9998766054240137]
```

The step from linear regression to logistic regression is kind of straightforward.

In the linear regression model, we have modelled the relationship between outcome and features with a linear equation:

y = b0 + b1x1 + b2x2 ....bnxn

For classification, we prefer probabilities between 0 and 1, so we wrap the right side of the equation into the logistic function.

$$y = \frac{1}{1 + e^{-b0 + b1x1 + b2x2 \,\ldots\ldots\ldots\, bnxn}}$$

In [10]:
```
#Feature Selection
x = data.iloc[:,:-1].values
y = data.iloc[:,-1].values
```

In [11]:
```
#split Data
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=1)
```

In [12]:
```
# Build model
lr = LogisticRegression()
lr.fit(x_train,y_train)
```

Out[12]:
```
LogisticRegression()
```

In [13]:
```
import pickle
Pkl_Filename = "students.pkl"
with open(Pkl_Filename, 'wb') as file:
    pickle.dump(lr, file)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [24]:
```
print (lr.intercept_, lr.coef_)   # b0 , b1 ,b2
```

```
[-11.80467167] [[1.2770634  1.05368774]]
```

In [55]:
```
y_pred = lr.predict(x_test)
```

In [56]:
```
# Evaluate Model
metrics.accuracy_score(y_test,y_pred)
```

Out[56]:
```
0.9
```

In [67]:
```
x_test
```

Out[67]:
```
array([[9.00037648, 9.54932786],
       [0.66547833, 9.78263644],
       [6.33309623, 7.24030498],
       [9.68310414, 9.50704973],
       [6.46089606, 7.07629269],
       [4.35755102, 9.88798331],
       [8.34775105, 1.86081251],
       [2.99191148, 5.29921046],
       [7.18081269, 3.61076342],
       [9.08162719, 1.4373504 ],
       [9.15051612, 2.56233373],
       [4.53017137, 3.761759  ],
       [7.69192306, 8.29822782],
       [0.90407766, 9.42092878],
       [5.9409696 , 4.62063163],
       [2.92268449, 8.21759492],
       [2.34361853, 2.95870685],
       [0.09805288, 7.21451254],
       [3.44310934, 7.06634686],
       [6.86140497, 9.65530971]])
```

In [66]:
```
y22 = lr.predict(x22)
y22
```

Out[66]:
```
array([1], dtype=int64)
```

In [69]:
```
from sklearn.datasets import load_iris
```

In [70]:
```
data1 = load_iris()
```

In [ ]:

In [5]:
```
data1.feature_names
```

Out[5]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [76]:
```
x1 = data1.data
y1 = data1.target
```

In [77]:
```
#split Data
x1_train,x1_test,y1_train,y1_test = train_test_split(x1,y1,test_size=0.2,random_state=1)
```

In [78]:
```
lr1 = LogisticRegression()
```

In [79]:
```
lr1.fit(x1_train,y1_train)
```

```
C:\Anaconda3\envs\daailab2020\lib\site-packages\sklearn\linear_model\_logistic.py:764: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/mo
dules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-
learn.org/stable/modules/linear_model.html#logistic-regression)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[79]:
```
LogisticRegression()
```

In [80]:
```
y1_pred = lr1.predict(x1_test)
```

In [81]:
```
# Evaluate Model
metrics.accuracy_score(y1_test,y1_pred)
```

Out[81]:
```
0.9666666666666667
```

In [82]:
```
y1_pred
```

Out[82]:
```
array([0, 1, 1, 0, 2, 1, 2, 0, 0, 2, 1, 0, 2, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       2, 0, 2, 1, 0, 0, 1, 2])
```

In [83]:
```
y1_test
```

Out[83]:
```
array([0, 1, 1, 0, 2, 1, 2, 0, 0, 2, 1, 0, 2, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 0, 2, 1, 0, 0, 1, 2])
```

Confusion Matrix

A confusion matrix is a summary of prediction results on a classification problem.

The number of correct and incorrect predictions are summarized with count values

In [13]:
```
from sklearn.metrics import confusion_matrix
```

In [28]:
```
confusion_matrix(y_test, y_pred)
```

Out[28]:
```
array([[ 5,  1],
       [ 1, 13]], dtype=int64)
```

In [29]:
```
18/20
```
Out[29]:
```
0.9
```

Get dummies

In [84]:
```
t_data = pd.read_csv('Titanic.csv')
t_data.head()
```
Out[84]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [85]:
```
pd.get_dummies(t_data['Sex'])
```
Out[85]:

	female	male
0	0	1
1	1	0
2	1	0
3	1	0
4	0	1
...	...	...
886	0	1
887	1	0
888	1	0
889	0	1
890	0	1

891 rows × 2 columns

In [87]:
```
pd.get_dummies(t_data['Sex'],drop_first=True)
```

Out[87]:

	male
0	1
1	0
2	0
3	0
4	1
...	...
886	1
887	0
888	0
889	1
890	1

891 rows × 1 columns

In [88]:
```
pd.get_dummies?
```

In [ ]: