Good Morning To All

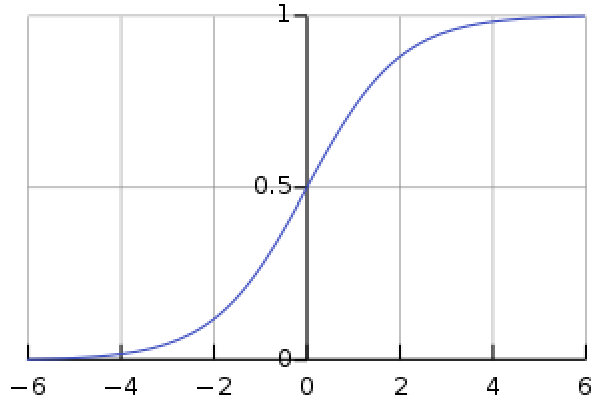
Logistic Regression

Decision Tree

Logistic Regression

Logistic Regression is a one of the Classification Algorithms

it works on the probability it uses logistic or Sigmoid Function



Now we can apply logistic Regression for Iris Dataset

In [2]:
```
import pandas as pd
```

In [3]:
```
from sklearn.datasets import load_iris
```

In [4]:
```
iris = load_iris()
```

In [5]:
```python
iris.keys()
```

Out[5]:
```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names'])
```

In [6]:
```python
iris['data']
```

Out[6]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
```

In [7]:
```python
iris['target_names']
```

Out[7]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [8]:
```python
iris['target']
```

Out[8]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [9]:
```
1  iris['DESCR']
```

Out[9]:
```
'Iris Plants Database\n====================\n\nNotes\n-----\nData Set Character
istics:\n    :Number of Instances: 150 (50 in each of three classes)\n    :Numb
er of Attributes: 4 numeric, predictive attributes and the class\n    :Attribut
e Information:\n        - sepal length in cm\n        - sepal width in cm\n
- petal length in cm\n        - petal width in cm\n        - class:\n
- Iris-Setosa\n                - Iris-Versicolour\n                - Iris-Virgi
nica\n    :Summary Statistics:\n\n    ============== ==== ==== ======= ===== ==
==================\n                    Min  Max   Mean    SD   Class Correlati
on\n    ============== ==== ==== ======= ===== ====================\n    sepal
length:   4.3  7.9   5.84   0.83    0.7826\n    sepal width:    2.0  4.4   3.05
0.43   -0.4194\n    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)\n
petal width:    0.1  2.5   1.20  0.76     0.9565  (high!)\n    ============== =
=== ==== ======= ===== ====================\n\n    :Missing Attribute Values: N
one\n    :Class Distribution: 33.3% for each of 3 classes.\n    :Creator: R.A.
Fisher\n    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n    :Date:
July, 1988\n\nThis is a copy of UCI ML iris datasets.\nhttp://archive.ics.uci.e
du/ml/datasets/Iris\n\nThe famous Iris database, first used by Sir R.A Fisher\n
\nThis is perhaps the best known database to be found in the\npattern recogniti
on literature.  Fisher\'s paper is a classic in the field and\nis referenced fr
equently to this day.  (See Duda & Hart, for example.)  The\ndata set contains
3 classes of 50 instances each, where each class refers to a\ntype of iris plan
t.  One class is linearly separable from the other 2; the\nlatter are NOT linea
rly separable from each other.\n\nReferences\n----------\n   - Fisher,R.A. "The
use of multiple measurements in taxonomic problems"\n     Annual Eugenics, 7, P
art II, 179-188 (1936); also in "Contributions to\n     Mathematical Statistic
s" (John Wiley, NY, 1950).\n   - Duda,R.O., & Hart,P.E. (1973) Pattern Classifi
cation and Scene Analysis.\n     (Q327.D83) John Wiley & Sons.  ISBN 0-471-2236
1-1.  See page 218.\n   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhoo
d: A New System\n     Structure and Classification Rule for Recognition in Part
ially Exposed\n     Environments".  IEEE Transactions on Pattern Analysis and M
achine\n     Intelligence, Vol. PAMI-2, No. 1, 67-71.\n   - Gates, G.W. (1972)
"The Reduced Nearest Neighbor Rule".  IEEE Transactions\n     on Information Th
eory, May 1972, 431-433.\n   - See also: 1988 MLC Proceedings, 54-64.  Cheesema
n et al"s AUTOCLASS II\n     conceptual clustering system finds 3 classes in th
e data.\n   - Many, many more ...\n'
```

In [10]:
```
1  iris['feature_names']
```

Out[10]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [12]:

```
df = pd.DataFrame(iris['data'])
df
```

Out[12]:

	0	1	2	3
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1
10	5.4	3.7	1.5	0.2
11	4.8	3.4	1.6	0.2
12	4.8	3.0	1.4	0.1
13	4.3	3.0	1.1	0.1
14	5.8	4.0	1.2	0.2
15	5.7	4.4	1.5	0.4
16	5.4	3.9	1.3	0.4
17	5.1	3.5	1.4	0.3
18	5.7	3.8	1.7	0.3
19	5.1	3.8	1.5	0.3
20	5.4	3.4	1.7	0.2
21	5.1	3.7	1.5	0.4
22	4.6	3.6	1.0	0.2
23	5.1	3.3	1.7	0.5
24	4.8	3.4	1.9	0.2
25	5.0	3.0	1.6	0.2
26	5.0	3.4	1.6	0.4
27	5.2	3.5	1.5	0.2
28	5.2	3.4	1.4	0.2
29	4.7	3.2	1.6	0.2
...	...	...	...	...
120	6.9	3.2	5.7	2.3
121	5.6	2.8	4.9	2.0

	0	1	2	3
122	7.7	2.8	6.7	2.0
123	6.3	2.7	4.9	1.8
124	6.7	3.3	5.7	2.1
125	7.2	3.2	6.0	1.8
126	6.2	2.8	4.8	1.8
127	6.1	3.0	4.9	1.8
128	6.4	2.8	5.6	2.1
129	7.2	3.0	5.8	1.6
130	7.4	2.8	6.1	1.9
131	7.9	3.8	6.4	2.0
132	6.4	2.8	5.6	2.2
133	6.3	2.8	5.1	1.5
134	6.1	2.6	5.6	1.4
135	7.7	3.0	6.1	2.3
136	6.3	3.4	5.6	2.4
137	6.4	3.1	5.5	1.8
138	6.0	3.0	4.8	1.8
139	6.9	3.1	5.4	2.1
140	6.7	3.1	5.6	2.4
141	6.9	3.1	5.1	2.3
142	5.8	2.7	5.1	1.9
143	6.8	3.2	5.9	2.3
144	6.7	3.3	5.7	2.5
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [13]:

```
df.columns = iris['feature_names']
df['target'] = iris['target']

```

In [14]:
```
df['target']
```

Out[14]:
```
0      0
1      0
2      0
3      0
4      0
5      0
6      0
7      0
8      0
9      0
10     0
11     0
12     0
13     0
14     0
15     0
16     0
17     0
18     0
19     0
20     0
21     0
22     0
23     0
24     0
25     0
26     0
27     0
28     0
29     0
      ..
120    2
121    2
122    2
123    2
124    2
125    2
126    2
127    2
128    2
129    2
130    2
131    2
132    2
133    2
134    2
135    2
136    2
137    2
138    2
139    2
140    2
141    2
142    2
```

```
143    2
144    2
145    2
146    2
147    2
148    2
149    2
Name: target, Length: 150, dtype: int32
```

In [19]:

```
df.sample(5)
```

Out[19]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
66	5.6	3.0	4.5	1.5	1
67	5.8	2.7	4.1	1.0	1
123	6.3	2.7	4.9	1.8	2
91	6.1	3.0	4.6	1.4	1
38	4.4	3.0	1.3	0.2	0

In [21]:
```
df['target']
```

Out[21]:
```
0      0
1      0
2      0
3      0
4      0
5      0
6      0
7      0
8      0
9      0
10     0
11     0
12     0
13     0
14     0
15     0
16     0
17     0
18     0
19     0
20     0
21     0
22     0
23     0
24     0
25     0
26     0
27     0
28     0
29     0
      ..
120    2
121    2
122    2
123    2
124    2
125    2
126    2
127    2
128    2
129    2
130    2
131    2
132    2
133    2
134    2
135    2
136    2
137    2
138    2
139    2
140    2
141    2
142    2
```

```
143    2
144    2
145    2
146    2
147    2
148    2
149    2
Name: target, Length: 150, dtype: int32
```

In [22]:
```
df['target'].unique()
```

Out[22]:
```
array([0, 1, 2], dtype=int64)
```

No of Sample in the data set

In [23]:
```
df.shape
```

Out[23]:
```
(150, 5)
```

In [24]:
```
df.isna().sum() # There is no missing values in this dataset
```

Out[24]:
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
target               0
dtype: int64
```

Is there any invalid values

1. all features should contain only numbers

2. target only contains integers

In [25]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal length (cm)    150 non-null float64
sepal width (cm)     150 non-null float64
petal length (cm)    150 non-null float64
petal width (cm)     150 non-null float64
target               150 non-null int32
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

In [26]:
```
df.corr()
```

Out[26]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
sepal length (cm)	1.000000	-0.109369	0.871754	0.817954	0.782561
sepal width (cm)	-0.109369	1.000000	-0.420516	-0.356544	-0.419446
petal length (cm)	0.871754	-0.420516	1.000000	0.962757	0.949043
petal width (cm)	0.817954	-0.356544	0.962757	1.000000	0.956464
target	0.782561	-0.419446	0.949043	0.956464	1.000000

sepal length (cm) petal length (cm) petal width (cm) hig correlation

In [29]:
```
X = df[['sepal length (cm)','petal length (cm)','petal width (cm)']]
y = df['target']
```

In [27]:
```
from sklearn.linear_model import LogisticRegression
```

In [30]:
```
logisticObject = LogisticRegression()
logisticObject.fit(X,y)
```

Out[30]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [31]:
```
logisticObject.score(X,y)
```

Out[31]:
```
0.9466666666666667
```

Error Metrics

In [32]:
```
from sklearn.metrics import confusion_matrix

```

In [33]:
```
y_actual = ['cat','ant','cat','cat','ant','bird']
y_pred = ['ant','ant','cat','cat','ant','cat']
```

--> left to right predicted and top to bottom actual

| ant bird cat

ant | 2 0 0

ant | 2 0 0
bird | 0 0 1

cat | 1 0 2

In [34]:
```
confusion_matrix(y_actual,y_pred,labels = ["ant","bird","cat"])
```

Out[34]:
```
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]], dtype=int64)
```

In [35]:
```
X.head(3)
```

Out[35]:

	sepal length (cm)	petal length (cm)	petal width (cm)
0	5.1	1.4	0.2
1	4.9	1.4	0.2
2	4.7	1.3	0.2

In [36]:
```
y_predict = logisticObject.predict(X)
y_predict
```

Out[36]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2,
       2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [37]:
```
confusion_matrix(y,y_predict)
```

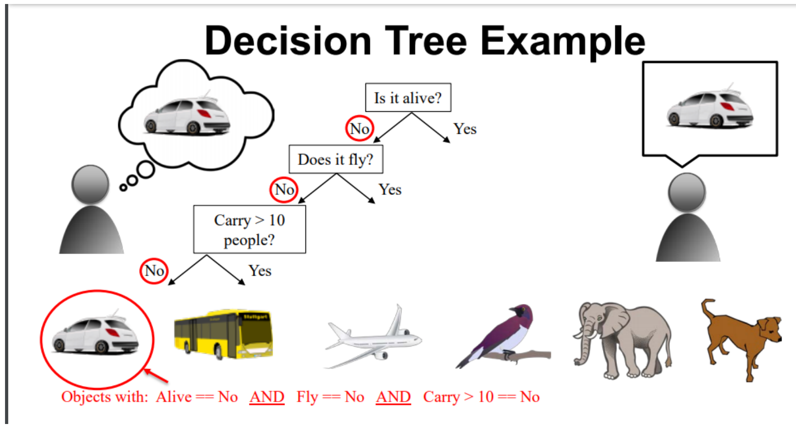
Out[37]:
```
array([[50,  0,  0],
       [ 0, 44,  6],
       [ 0,  2, 48]], dtype=int64)
```

In [38]:
```
from sklearn.metrics import accuracy_score
```

In [39]:
```
accuracy_score(y,y_predict)
```

Out[39]:
```
0.9466666666666667
```

Decision Tree Classifica



In [40]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [41]:
```
from sklearn.model_selection import train_test_split
```

In [42]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.7)
print(X_train.shape)
```

```
(105, 3)

C:\Users\ravi sastry\Anaconda3\lib\site-packages\sklearn\model_selection\_spli
t.py:2026: FutureWarning: From version 0.21, test_size will always complement
train_size unless both are specified.
  FutureWarning)
```

In [43]:
```
dtree = DecisionTreeClassifier()
dtree.fit(X_train,y_train)
print(dtree.score(X_train,y_train))
dtree.score(X_test,y_test)



```

```
1.0
```

Out[43]:
```
0.9333333333333333
```

In [44]:
```
confusion_matrix(y_train,dtree.predict(X_train))
```

Out[44]:
```
array([[33,  0,  0],
       [ 0, 33,  0],
       [ 0,  0, 39]], dtype=int64)
```

In [45]:
```
aconfusion_matrix(y_test,dtree.predict(X_test))
```

Out[45]:
```
array([[17,  0,  0],
       [ 0, 15,  2],
       [ 0,  1, 10]], dtype=int64)
```

In [46]:
```
accuracy_score(y_train,dtree.predict(X_train))
```
Out[46]:
```
1.0
```
In [47]:
```
accuracy_score(y_test,dtree.predict(X_test))
```
Out[47]:
```
0.9333333333333333
```
In [ ]:
```

```