Implementation of Mean Absolute Error, Mean Squared Error & Log Loss Error

Mean Absolute Error Formula

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

In [12]:
```
import numpy as np

y_predicted=np.array([1,1,0,0,1])                          # Y-hat
y_true=np.array([0.30,0.7,1.0,0.5,1])                      # y
```

In [51]:
```
def mean_absolute_error(y_true,y_predicated):
    total_error=0
    for yt,yp in zip(y_true,y_predicted):       #zip fun is used to interate two variables y_true & y_predicted c
        total_error+=abs(yt-yp)
        print('Total Error:',total_error)
        mae=total_error/len(y_true)
        print('MAE:',mae)

        return mae
```

In [52]:
```
mean_absolute_error(y_true,y_predicted)
```

```
Total Error: 0.7
MAE: 0.13999999999999999
```

Out[52]:
```
0.13999999999999999
```

In [54]:
```python
np.mean(np.abs(y_predicted - y_true))  # the same can be implemented using numpy function np.abs()
```

Out[54]:
```
0.5
```

MSE formula

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

In [55]:
```python
y_predicted=np.array([1,1,0,0,1])                    # Y-hat
y_true=np.array([0.30,0.7,1.0,0.5,1])                # y
```

In [62]:
```python
def mean_squared_error(y_true,y_predicted):
    total_error=0
    for yt,yp in zip(y_true,y_predicted):
        total_error+=np.square(yt-yp)
        print('Total Error:', total_error)
        mse=total_error/len(y_true)
        print('MSE:',mse)
        return mse
```

In [64]:
```python
mean_squared_error(y_true,y_predicted)
```

```
Total Error: 0.48999999999999994
MSE: 0.09799999999999999
```

Out[64]:
```
0.09799999999999999
```

In [66]:
```
np.mean(np.square(np.subtract(y_true,y_predicted)))
```
Out[66]:
```
0.366
```

Log Loss formula

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1-y_i) \cdot log(1-p(y_i))$$

Binary Cross-Entropy / Log Loss

In [23]:
```
np.log(0.0000000000000000001)
```
Out[23]:
```
-41.44653167389282
```

In [24]:
```
epsilon=1e-15
```

In [25]:
```
y_predicted
```
Out[25]:
```
array([1, 1, 0, 0, 1])
```

In [32]:
```
y_predicted_new=[max(i,epsilon) for i in y_predicted]
y_predicted_new
```
Out[32]:
```
[1, 1, 1e-15, 1e-15, 1]
```

In [33]:
```
y_predicted_new=[min(i,1-epsilon) for i in y_predicted_new]
y_predicted_new
```

Out[33]: [0.999999999999999, 0.999999999999999, 1e-15, 1e-15, 0.999999999999999]

In [34]:
```
y_predicted_new=np.array(y_predicted_new)
np.log(y_predicted_new)
```

Out[34]:
```
array([-9.99200722e-16, -9.99200722e-16, -3.45387764e+01, -3.45387764e+01,
       -9.99200722e-16])
```

In [37]:
```
-np.mean(y_true*np.log(y_predicted_new)+(1-y_true)*np.log(1-y_predicted_new))
```

Out[37]: 17.26954811694138

In [44]:
```
def log_loss(y_true,y_predicted):
    epsilon= 1e-15
    y_predicted_new=[max(i,epsilon) for i in y_predicted]
    y_predicted_new=[min(i,1-epsilon) for i in y_predicted_new]
    y_predicted_new= np.array(y_predicted_new)
    return -np.mean(y_true*np.log(y_predicted_new)+(1-y_true)*np.log(1-y_predicted_new))
```

In [45]:
```
log_loss(y_true,y_predicted)
```

Out[45]: 17.26954811694138

In [ ]: