Content

- Entropy for Continuous Random Variable
- Comparing Entropy for Continuous R.V
- Code walkthrough of splitting numerical features

Formulation - Continuous R.V

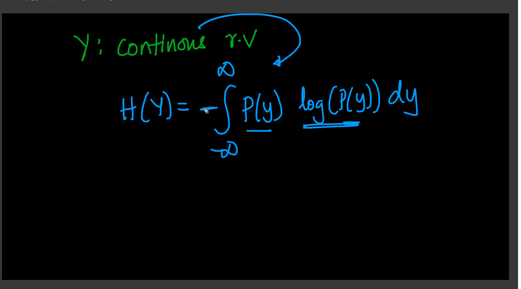
In discrete random variable, we use summation over all values.

When we have continous RV, this summation turns into integration.

The forumlation becomes

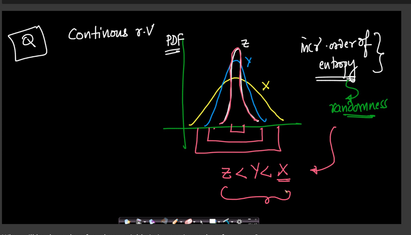
$H(Y) = -\int_{-\infty}^{\infty} P(y) log(P(y)) dy$

where p(y) is probability density function



Comparing entropy for continuous R.V

```
Say we have 3 continous random variables X, Y, Z
and their distribution (PFF) is as follows:
```



What will be the order of random variable in increasing order of entropy ?

Z < Y < X

Remember that entropy is the measure of randomness.

- Here, Z has a peaked curve
  - We are more likely to observe values in very small range (low variance)
  - Means there is more certainity and less randomness
  - Hence, Z will have lowest entropy
- Random Variable Y has a large spread compared to Z
  - we are more likely to observe value from a wider range
  - meaning, there is more randomness in values
  - Hence, it'll have high entropy compared to Z

In similar fashion, X having the largest spread will have maximum entropy

More the variance, more the entropy

So, the order becomes Z < Y < X

Code walkthrough of Splitting numerical features

Importing libraries

```
[ ]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import io
```

Preprocessed Data

```
[ ]  !gdown 12G_J8qBel8MpspdXJruZuuMvyrm9upNb
     !gdown 1-zjfYbZm8JC4wog9tG4zltMPuZM6p9_A


     !gdown 1p03LtdtoBXoOdu14YlcQjzE-360SyI89
     !gdown 1mNcWfaEW19ql8WCihGHiqRryCcjFUUqx

     !gdown 1ZoA7Vu1V48nPafCNJSqhG8yVvyFfC6OA
```

```
     Downloading...
     From: https://drive.google.com/uc?id=12G_J8qBel8MpspdXJruZuuMvyrm9upNb
     To: /content/X_test.csv
     100% 51.7k/51.7k [00:00<00:00, 88.5MB/s]
     Downloading...
     From: https://drive.google.com/uc?id=1-zjfYbZm8JC4wog9tG4zltMPuZM6p9_A
     To: /content/X_train.csv
     100% 154k/154k [00:00<00:00, 115MB/s]
     Downloading...
     From: https://drive.google.com/uc?id=1p03LtdtoBXoOdu14YlcQjzE-360SyI89
     To: /content/y_test.csv
     100% 743/743 [00:00<00:00, 1.99MB/s]
     Downloading...
     From: https://drive.google.com/uc?id=1mNcWfaEW19ql8WCihGHiqRryCcjFUUqx
     To: /content/y_train.csv
     100% 2.21k/2.21k [00:00<00:00, 4.85MB/s]
     Downloading...
     From: https://drive.google.com/uc?id=1ZoA7Vu1V48nPafCNJSqhG8yVvyFfC6OA
     To: /content/target.csv
     100% 2.95k/2.95k [00:00<00:00, 8.18MB/s]
```

```
[ ]  X_train = pd.read_csv('X_train.csv')
     X_test = pd.read_csv('X_test.csv')
     y_train = pd.read_csv('y_train.csv')
     y_test = pd.read_csv('y_test.csv')

     target  = pd.read_csv('target.csv')
     target = target.iloc[:,0]
```

```
[ ]  from imblearn.over_sampling import SMOTE
     from collections import Counter

     smt = SMOTE()
     X_sm, y_sm = smt.fit_resample(X_train, y_train)

     print('Resampled dataset shape {}'.format(Counter(y_sm)))
```

```
     Resampled dataset shape Counter({'Target': 1})
```

Let's split the Age feature and find which threshold is best to split age along with its information gain

```
[ ]  age = X_sm.Age
```

Sorting the age

```
[ ]  thresholds = age.sort_values().unique()
     thresholds
```

```
     array([18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
            35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
            52, 53, 54, 55, 56, 57, 58, 59, 60])
```

```
[ ]  thresholds.shape
```

```
     (43,)
```

Calculating information gain for each threshold

```
[ ]  def gini_impurity(y):

       if isinstance(y, pd.Series):
         p = y.value_counts()/y.shape[0]
         gini = 1-np.sum(p**2)
         return gini

       else:
         raise('Object must be a Pandas Series.')
```

```
[ ]  def information_gain(y, mask):
         left_node_count = sum(mask)
         total = mask.shape[0]
         right_node_count = total - left_node_count


         parent_gini = gini_impurity(y)

         child_gini = left_node_count/total*gini_impurity(y[mask]) + right_node_count/total*gini_impurity(y[~mask])

         ig = parent_gini - child_gini
         return ig
```

```
[ ]  ig_list = []

     for thr in thresholds:
         mask =   age <= thr
         ig = information_gain(target, mask)
         ig_list.append(ig)
```

```
[ ]  ig_list = np.array(ig_list)

     ig_list.shape
```

```
     (43,)
```

Finding threshold with maximum IG

```
[ ]  print(f'Best threshold for Age with maximum IG is {thresholds[ig_list.argmax()]} with IG: {ig_list.max()}')
```

```
     Best threshold for Age with maximum IG is 46 with IG: 0.0029346385275854647
```