Lab : Communication Engineering by Python

Created by	Ⓑ Borhan
Last edited time	@January 7, 2025 11:24 PM
Tag	Year 3 Term 1

Theoretical Concepts

Entropy:

$$H(X) = -\sum P(x_i) \log_2 P(x_i)$$

Joint Entropy:

$$H(X,Y) = -\sum_{x \epsilon X} \sum_{y \epsilon Y} P(x,y) \log_2 P(x,y)$$

Mutual Information

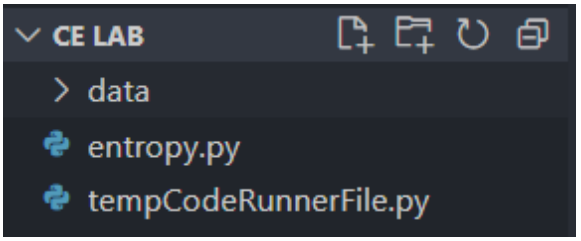
$$M(X,Y) = H(X) + H(Y) - H(X,Y)$$

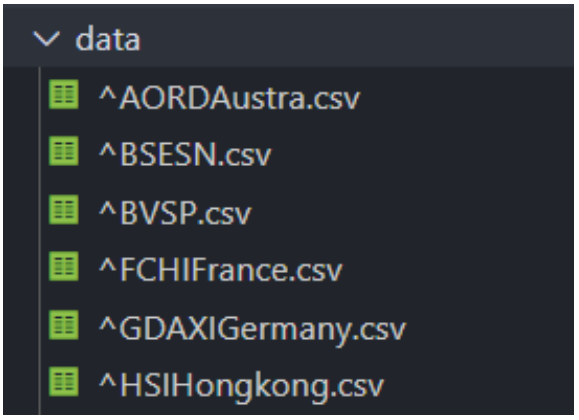
Packages

- Pandas
- NumPy
- SciPy

```
pip install pandas numpy scipy
```

My File Structure





Code

Reading a CSV file

```
import pandas as pd

data = pd.read_csv("data/^TWII.csv");
print(data.head());
```

Calculating Information Content

```
import pandas as pd
import numpy as np

data = pd.read_csv("data/^TWII.csv")
column = data['Open']
value, counts = np.unique(column, return_counts=True)

prob = counts / len(column)

for i in range(len(value)) :
    information = -np.log2(prob[i])
    print("Symbol: ", value[i], "Information: ", information)
```

Calculating Entropy

```
import pandas as pd
import numpy as np

data = pd.read_csv("data/^TWII.csv")
data['Date'] = pd.to_datetime(data['Date'], format='%Y-%m-%d'
column = data['Open']
column = column.dropna()
values, counts = np.unique(column, return_counts=True)
counts = counts[np.nonzero(counts)]

prob = []

for i in range(len(counts)):
    prob.append(counts[i] / len(column));
# Alternative : prob = counts / len(column)

def entropy(prob) :
    entropy = 0
    for i in range(len(prob)):
        entropy -= prob[i] * np.log2(prob[i]);
    return entropy

print(entropy(prob))
```

Calculating Entropy by Using Bins

```
import pandas as pd
import numpy as np

data = pd.read_csv("data/canada.csv")
data['Date'] = pd.to_datetime(data['Date'], format='%Y-%m-%d'
data.sort_values(by='Date', ascending=True, inplace=True)
```

```
column = data['Close']
column= np.log(data['Close']) - np.log(data['Close'].shift(1)
column = column.dropna()

print(f"Max Value : {column.max()}")
print(f"Min Value : {column.min()}")
print(f"Differece : {column.max() - column.min()}")
print("Bins : 12")

def entropy(column, bins) :
    binned_dist, bin_edges = np.histogram(column, bins)
    probs = binned_dist / np.sum(binned_dist)
    bin_table = pd.DataFrame({
        'Bin Range': [f"[{bin_edges[i]:.4f}, {bin_edges[i+1]:
        'Frequency': binned_dist,
        'Probability': probs
    });
    print(bin_table)
    entropy = 0
    for i in range(len(probs)):
        entropy -= probs[i] * np.log2(probs[i]);
    return entropy

print(f"Entropy : {entropy(column, 12)}")
```

NOTE :

```
    for i in range(len(prob)):
        entropy -= prob[i] * np.log2(prob[i]);

    This portion may be written,

    entropy = -np.sum(probs * np.log2(probs));
```

Mutual Information

```
import pandas as pd
import numpy as np

# X
data = pd.read_csv("data/canada.csv");
X = data[['Date', 'Close']];
X.loc[:, 'Date'] = pd.to_datetime(X['Date'])
X.loc[:, 'Close'] = np.log(X['Close']) - np.log(X['Close'].sh
X = X[X['Close'].notna()];
X.sort_values(by='Date', ascending=True, inplace=True)
X = X.dropna()

# Y
data2 = pd.read_csv("data/^TWII.csv");
Y = data2[['Date', 'Close']];
Y.loc[:, 'Date'] = pd.to_datetime(Y['Date'])
Y.loc[:, 'Close']  = np.log(Y['Close']) - np.log(Y['Close'].s
Y = Y[Y['Close'].notna()];
Y.sort_values(by='Date', ascending=True, inplace=True)
Y = Y.dropna()

# Merge
XY = pd.merge(X, Y, on='Date', how='inner');



def entropy(X, bins):
    print(f"Max Value : {X.max() : .4f}")
    print(f"Min Value : {X.min() : .4f}")
    print(f"Differece : {X.max() - X.min() : .4f}")
    print(f"Bins : {bins}")
    binned_dist, bin_edges = np.histogram(X, bins)
    probs = binned_dist / np.sum(binned_dist)


    bin_table = pd.DataFrame({
        'Bin Range': [f"[{bin_edges[i]:.4f}, {bin_edges[i+1]:
        for i in range(len(bin_edges)-1)],
```

```
        'Frequency': binned_dist,
        'Probability': probs
    });
    print(bin_table)
    probs = probs[np.nonzero(probs)]
    entropy = -np.sum(probs * np.log2(probs))

    return entropy

def jointEntropy(X, Y, bins):
    binned_XY, binnedX, binnedY = np.histogram2d(X, Y, bins)
    probsXY = binned_XY / np.sum(binned_XY)

    # Show the joint distribution
    rows = []
    for i in range(len(binnedX) - 1):
        for j in range(len(binnedY) - 1):
            rows.append({
                'X Bin': f"[{binnedX[i]:.4f}, {binnedX[i+1]:.
                'Y Bin': f"[{binnedY[j]:.4f}, {binnedY[j+1]:.
                'Frequency': binned_XY[i, j],
                'Probability': probsXY[i, j]
            })

    table = pd.DataFrame(rows)
    print(table)

    probsXY = probsXY[np.nonzero(probsXY)]
    jointEntropy = -np.sum(probsXY * np.log2(probsXY))

    return jointEntropy

def mutualInformation(X, Y, bins) :
    print("X : ");
    entropyX = entropy(X, bins);
    print(f"Entropy X, H(X) : {entropyX:.4f}");
    print("\n\nY : ");
    entropyY = entropy(Y, bins);
```

```
    print(f"Entropy Y, H(Y) : {entropyY:.4f}");
    print("\nJoint : ");
    joint = jointEntropy(X, Y, bins);
    print(f"Joint Entropy H(X, Y) : {joint:.4f}");
    return entropyX + entropyY - joint

print(f"\n Mutual Information: {mutualInformation(XY['Close_x
: .4f}");
```

Using Library function

```
import pandas as pd
import numpy as np
from scipy.stats import entropy
from sklearn.metrics import mutual_info_score

data = pd.read_csv("data/^TWII.csv")
bins = 100;

#X
X = data['Open']
X = X.dropna()
X_binned = np.histogram(X, bins)[0];

#Y
Y = data['Close']
Y = Y.dropna()
Y_binned = np.histogram(Y, bins)[0];

print(entropy(X_binned, base=2))
print(entropy(Y_binned, base=2))

mutual_info = mutual_info_score(X_binned,Y_binned)
print(mutual_info)
```