Association Rule example

https://stackabuse.com/association-rule-mining-via-apriori-algorithm-in-python/

https://medium.com/analytics-vidhya/association-analysis-in-python-2b955d0180c

Simple Linear Regression

https://towardsdatascience.com/simple-linear-regression-in-python-numpy-only-130a988c0212

Classifications

https://towardsdatascience.com/solving-a-simple-classification-problem-with-python-fruits-lovers-edition-d20ab6b071d2

Clustering

https://towardsdatascience.com/machine-learning-algorithms-part-9-k-means-example-in-python-f2ad05ed5203

Activity 1 :

Dataset: http://archive.ics.uci.edu/ml/datasets/Online+Retail

Step 1: Importing the required libraries

Step 2: Loading and exploring the data

```
 # Loading the Data

data = pd.read_excel('Online_Retail.xlsx')

data.head()
```

```
 # Exploring the columns of the data

 data.columns
```

```
  # Stripping extra spaces in the description

  data['Description'] = data['Description'].str.strip()
```

```
  # Dropping the rows without any invoice number

  data.dropna(axis = 0, subset =['InvoiceNo'], inplace = True)

  data['InvoiceNo'] = data['InvoiceNo'].astype('str')
```

```
  # Dropping all transactions which were done on credit
```

```
data = data[~data['InvoiceNo'].str.contains('C')]
```

Step 3: Cleaning the Data

```
# Stripping extra spaces in the description

data['Description'] = data['Description'].str.strip()



# Dropping the rows without any invoice number

data.dropna(axis = 0, subset =['InvoiceNo'], inplace = True)

data['InvoiceNo'] = data['InvoiceNo'].astype('str')



# Dropping all transactions which were done on credit

data = data[~data['InvoiceNo'].str.contains('C')]
```

Step 4: Splitting the data according to the region of transaction

```
# Transactions done in France

basket_France = (data[data['Country'] =="France"]

                 .groupby(['InvoiceNo', 'Description'])['Quantity']

                 .sum().unstack().reset_index().fillna(0)

                 .set_index('InvoiceNo'))
```

```
# Transactions done in the United Kingdom
basket_UK = (data[data['Country'] =="United Kingdom"]
                    .groupby(['InvoiceNo', 'Description'])['Quantity']
                    .sum().unstack().reset_index().fillna(0)
                    .set_index('InvoiceNo'))

# Transactions done in Portugal
basket_Por = (data[data['Country'] =="Portugal"]
                    .groupby(['InvoiceNo', 'Description'])['Quantity']
                    .sum().unstack().reset_index().fillna(0)
                    .set_index('InvoiceNo'))

basket_Sweden = (data[data['Country'] =="Sweden"]
                    .groupby(['InvoiceNo', 'Description'])['Quantity']
                    .sum().unstack().reset_index().fillna(0)
                    .set_index('InvoiceNo'))
```

Step 5: Hot encoding the Data

```
# Defining the hot encoding function to make the data suitable

# for the concerned libraries

def hot_encode(x):

    if(x<= 0):

        return 0

    if(x>= 1):

        return 1



# Encoding the datasets

basket_encoded = basket_France.applymap(hot_encode)

basket_France = basket_encoded



basket_encoded = basket_UK.applymap(hot_encode)

basket_UK = basket_encoded
```

basket_encoded = basket_Por.applymap(hot_encode)

basket_Por = basket_encoded

basket_encoded = basket_Sweden.applymap(hot_encode)

basket_Sweden = basket_encoded

Step 6: Building the models and analyzing the results

a) **France:**

b) **United Kingdom:**

c) **Portugal:**

d) **Sweden:**

Activity 02 :

Apply the simple linear regression to below small data set

```
x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])
```

expected output :

Estimated coefficients:

b_0 = -0.0586206896552

b_1 = 1.45747126437

Activity 03:

Apply multiple linear regression to bostan dataset and get the output as below

```
# load the boston dataset
boston = datasets.load_boston(return_X_y=False)
```

Expected Output :

Coefficients:

[-8.80740828e-02 6.72507352e-02 5.10280463e-02 2.18879172e+00

-1.72283734e+01 3.62985243e+00 2.13933641e-03 -1.36531300e+00

2.88788067e-01 -1.22618657e-02 -8.36014969e-01 9.53058061e-03

-5.05036163e-01]

Variance score: 0.720898784611