In [9]:
```
import pandas as pd
from scipy.stats.stats import mode
from sklearn.model_selection import train_test_split
import numpy as np
# from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Perceptron
from sklearn.neighbors import KNeighborsClassifier
# from mlxtend.evaluate import bias_variance_decomp

#กำหนดว่าน่าซื้อหรือไม่
def range_price(price,avg):
    if(price<=avg):
        return 0
    elif(price>avg):
        return 1

#ลองเทสกับ Perceptron algorithm
def perceptron(x_train,y_train,x_test,y_test,input_value):
    a=Perceptron()
    a.fit(x_train[:,np.newaxis],y_train[:,np.newaxis])
    score=a.score(x_test[:,np.newaxis],y_test[:,np.newaxis])
    pre=a.predict(np.array([input_value])[:,np.newaxis])
    # print(score)
    return pre,score

#ลองเทสกับ  K-nearest neighbors algorithm
def knn(x_train,y_train,x_test,y_test,input_value):
    a=KNeighborsClassifier()
    a.fit(x_train[:,np.newaxis],y_train[:,np.newaxis])
    score=a.score(x_test[:,np.newaxis],y_test[:,np.newaxis])
    pre=a.predict(np.array([input_value])[:,np.newaxis])
    # print(score)
    return pre,score
```

In [10]:
```
#รับข้อมูลจาก data set
df=pd.read_csv('noit11561118811.csv')
type_pro=df['PR_PROD_NAME']
date=pd.DatetimeIndex(df['PRICE_DATE']).month

#เก็บประเภท
keep_type=[]
count=1
for type_name in type_pro:
    if(count==1):
        keep_type.append(type_name)
        count+=1
    if(type_name in keep_type):
        pass
    else:
        keep_type.append(type_name)
        count+=1
```

In [11]:
```
#เก็บค่าเฉลี่ยแต่ละปี
month=[1,2,3,4,5,6,7,8,9,10,11,12]
total={word:{mon:0 for mon in month} for word in keep_type}
length={word:{mon:0 for mon in month} for word in keep_type}
avg={word:{mon:0 for mon in month} for word in keep_type}
for keep in keep_type:
    for index,tyn in enumerate(type_pro):
        if(keep==tyn):
            total[keep][date[index]]+=df['PRICE_DAY'][index]
            length[keep][date[index]]+=1

for keep in keep_type:
    for index in month:
        if(length[keep][index]!=0):
            # print(keep)
            # print(length[keep][index])
            # print(index)
            avg[keep][index]=total[keep][index]/length[keep][index]
            # print(avg[keep][index])
```

In [12]:
```
#การทำนาย
def perdict_price(input_name,input_month,input_pre):
    # print(int(input_month))
    n=7
    input_month=np.int64(input_month)
    if(input_name in keep_type):
        if(input_month in month):
            # print(length[input_name][input_month])
            if(length[input_name][input_month]>=n):
                # print("AAA")
                keep_price=[]
                keep_rang=[]
                for index,tyn in enumerate(type_pro):
                    # print(input_name is tyn)
                    if(input_name == tyn):
                        # print(tyn)
                        # print(type(date[index]))
                        # print(date[index])
                        # print(type(input_month))
                        # print(input_month==date[index])
                        if(input_month==date[index]):
                            # print(date[index])
                            price=df['PRICE_DAY'][index]
                            keep_price.append(price)
                            rang=range_price(price,avg[input_name][date[index]])
                            keep_rang.append(rang)
                x=np.array(keep_price)
                y=np.array(keep_rang)
                x_train,x_test,y_train,y_test=train_test_split(x,y)
                if(len(x_train)<n and len(y_train)<n):
                  return [4],-1
                else:
                    pre,score=knn(x_train,y_train,x_test,y_test,input_pre)
                # print(pre)


                # mse,bias,var=bias_variance_decomp(KNeighborsClassifier,x_train[:,np.newaxis],y_tra
                #  num_rounds=200, random_seed=1)
                x_train,x_va,y_train,y_va=train_test_split(x_train,y_train)
                if(len(x_train)<n and len(y_train)<n):
                    return [4],-1
                else:
                    pre,score=knn(x_train,y_train,x_va,y_va,input_pre)
                # print(pre)
                x_train,x_va,y_train,y_va=train_test_split(x_train,y_train)
                if(len(x_train)<n and len(y_train)<n):
                    return [4],-1
                else:
                    pre,score=knn(x_train,y_train,x_va,y_va,input_pre)
                return pre,score
            else:
                return [4],-1
        else:
            return [3],-1
    else:
        return [2],-1
```

In [13]:
```
#คืนค่าของชื่อสินค้าทั้งหมด
def getType():
    return keep_type
#คืนค่าเฉลี่ยของสินค้านั้น
def getavg(input_name,input_month):
    return avg[input_name][input_month]
```

In [14]:
```
pre,score=perdict_price("ไข่ไก่เบอร์ 4",12,300)
print("Perdict:",pre)
print("Score:",score)
```

```
Perdict: [1]
Score: 1.0

C:\Users\PPMP\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:179: DataConve
rsionWarning: A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
C:\Users\PPMP\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:179: DataConve
rsionWarning: A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
C:\Users\PPMP\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:179: DataConve
rsionWarning: A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
```

In [ ]: