ECON 412 Project 1

Siyan Chen, Zeyi Pan, Minxuan Wang, Huafeng Zhang

UCLA Spring 2018

1 The Goal

In machine learning, a probabilistic classifier is a classifier that is able to predict, given an observation of an input, a probability distribution over a set of classes, rather than only outputting the most likely class that the observation should belong to. Probabilistic classifiers provide classification that can be useful in its own right or when combining classifiers into ensembles. For this project, our goal is to implement a naive Bayesian learning algorithm to the economic data set and to learn how the classified characteristics (e.g. age and education) of an individual can determine his/her hourly wage level (low, medium, or high) in a given metropolitan area in the early years last century.

For the second part, we are interested in studying how to compute and interpret the causal relations among data groups based upon Bayesian theory. Prior to that, we would also like to learn how the Kalman Filter works to fit the real data and provide optimal linear quadratic estimation of our observations. Then, we want to generate a Bayesian causal impact analysis on the data groups.

2 Data Acquisition

2.1 CPS Data for Naive Bayesian Classification

Our data source for the naive Bayesian learning part is the Current Population Surveys (CPS) from National Bureau of Economic Research (NBER) [1]. It offers the annual Merged Outgoing Rotation Groups survey information from 1979 to 2017 for each metropolitan area in the United States.

We use the data set of 1979 and take subsets of employed labor force observations in **New York, Los Angeles, and Miami**. Then we choose **race, gender, class of worker, age, and education level** as features for classification and use the features to determine the hourly

[1]http://www.nber.org/data/morg.html

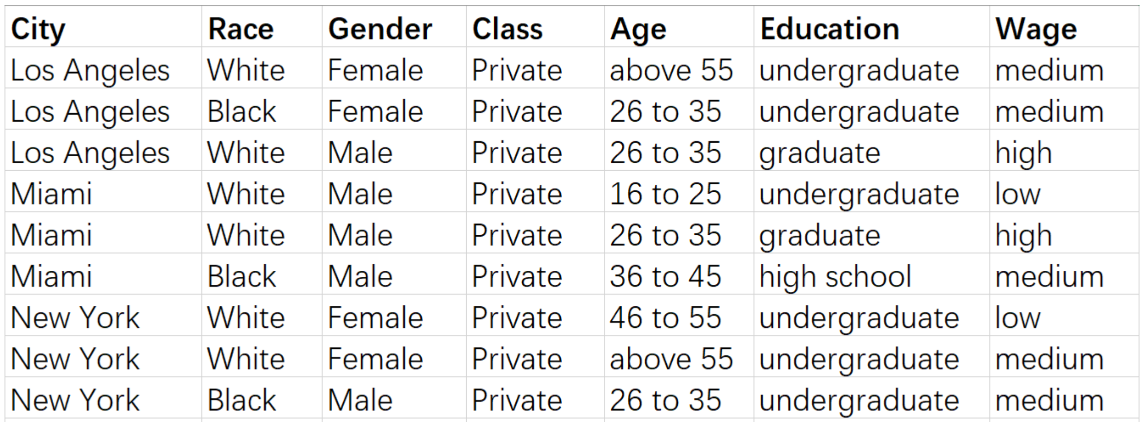
wage category.

Class of worker includes Private, Government, Self-Employed and others. We also classify the chosen variables age, education level and hourly wage.

Age of worker is divided to 6 categories: under 16, 16 to 25, 26 to 35, 36 to 45, 46 to 55 and above 55.

Education level is determined by the highest grade attended and divided into 3 categories: high school, undergraduate and graduate.

Hourly wage is divided into 3 categories: low level for hourly wages under 4 dollars, medium level for wages between 4 dollars and 10 dollar, and high level for wages over 10 dollars per hour.



City	Race	Gender	Class	Age	Education	Wage
Los Angeles	White	Female	Private	above 55	undergraduate	medium
Los Angeles	Black	Female	Private	26 to 35	undergraduate	medium
Los Angeles	White	Male	Private	26 to 35	graduate	high
Miami	White	Male	Private	16 to 25	undergraduate	low
Miami	White	Male	Private	26 to 35	graduate	high
Miami	Black	Male	Private	36 to 45	high school	medium
New York	White	Female	Private	46 to 55	undergraduate	low
New York	White	Female	Private	above 55	undergraduate	medium
New York	Black	Male	Private	26 to 35	undergraduate	medium

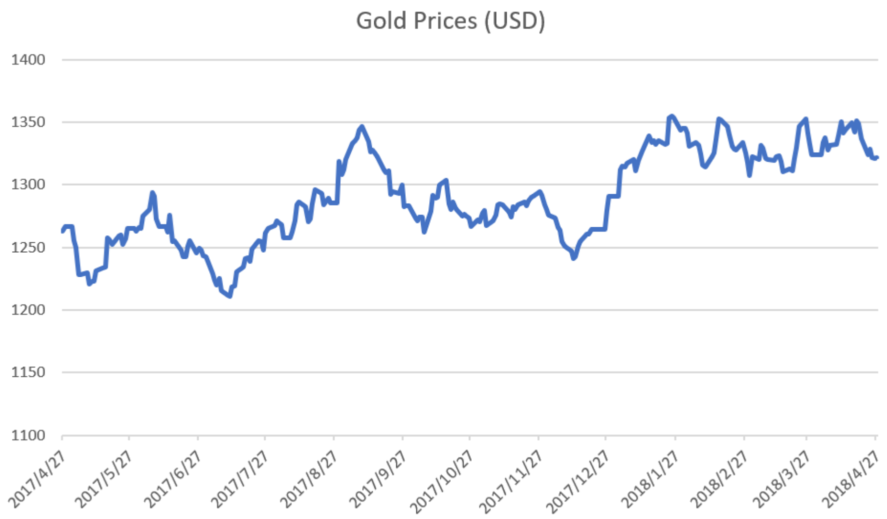
This is a **subset sample** of our data set, we have **5287** data points in total (different individuals) as training set.

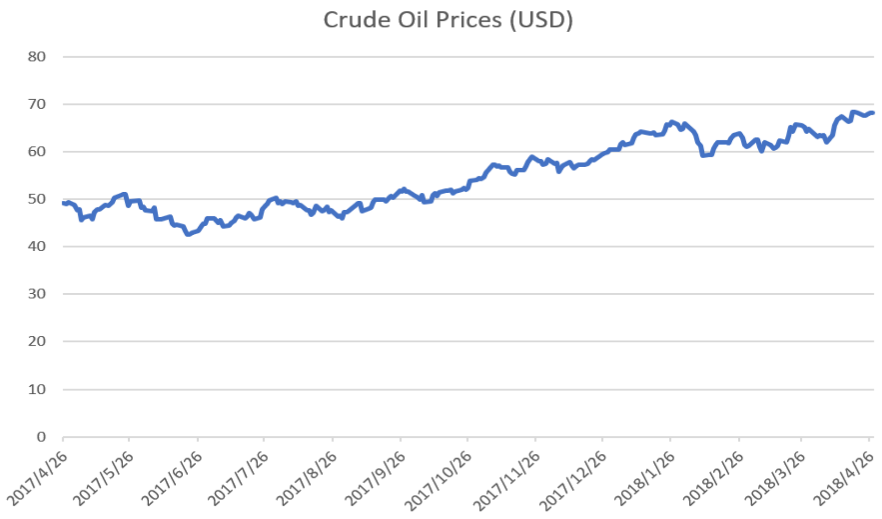
2.2 Gold and Crude oil Prices for Causal Impact

From Quandl, we imported the daily gold prices [2] and OPEC crude oil prices[3] (in USD) for the recent 1 year (begin with $04/27/2017$) to R and applied particle filters to analyze the data. For each group, a total of 254 observations are available here because there are no updates for weekends or holidays. The trend plots of original data are presented below:

[2]https://www.quandl.com/data/WGC/GOLD_DAILY_USD-Gold-Prices-Daily-Currency-USD

[3]https://www.quandl.com/data/OPEC/ORB-OPEC-Crude-Oil-Price





3 Analytic Methods

3.1 Naïve Bayes Classifier

In terms of statistical tool, we first use R to deal with the data cleaning process and export the collated data set to a csv file for future use. Then, by importing the csv file to Python, we build a naïve Bayes classifier to conduct the classification. The goal of the classifier is to determine the wage level of an unknown sample with attributed features $(X_1, ..., X_6)$. We calculate the probability of the giving wage level in the entire current sample as our prior knowledge. Then, we find the posterior probability for different features. With the posterior probability updating our prior knowledge, we are able to generate the probability of the wage level for the observation with its given characteristics. At last, as we attribute certain characteristics to a test sample, we are able to classify the wage level of this sample as the class (low, medium, or high) based on the highest probability calculated. In formula, if $P(C = 1|X_1, ..., X_6) < P(C = 2|X_1, ..., X_6) < P(C = 3|X_1, ..., X_6)$, where $C = 1, 2, 3$ are different classes, we can conclude here that the current sample supports the unknown sample to be classified as C_3.

Bayes Theorem

$$P\left(C|X\right) = \frac{P\left(C|X\right)P\left(C\right)}{P\left(X\right)}$$

$P\left(C|X\right)$ is the posterior probability of C given the attribute X. $P\left(C\right)$ is the prior probability of C, and this formula is called ”Bayes theorem”. According to which, we want to find the largest $P\left(C|X\right)$. Because $P\left(X\right)$ is constant for all classes, we can achieve that as long as the maximum $P\left(X|C\right)P\left(C\right)$ is found, this is the cornerstone of Naive Bayes classification.

Naïve Bayes classification

Using the Bayes theorem, the largest $P\left(X|C\right)P\left(C\right)$ can be used to identify the classification of unknown samples, such as $maxP\left(X|C\right)P\left(C\right) = P\left(X|C = n\right)P\left(C = n\right)$, indicating that the unknown samples belong to the category n, in which:

(1) $P\left(C = i\right) = S_i/S$, where S_i is the number of training samples in class C_i, and S is the total number of training samples.

(2) For $P\left(X|C = i\right)$, as many attribute variables are involved, we assume that they are conditional independent, that is, there is no dependency between different attributes:

$$P\left(X|C = i\right) = \prod_{k=1}^{t} p\left(X_k|C = i\right)$$

That is, if six attribute values $X_1, \ldots, X_6$ are used to speculate on class C, there exists:

$$P(X_1, \ldots, X_3|C=i) = P(X_1|C=i) \times \ldots \times P(X_6|C=i)$$

This assumption is to simplify the computation, and therefore that's why this algorithm is called ”naive”.

(3) It should be noted that when calculating $P(X_k|C=i)$, it depends on whether X_k is categorized or continuous. If X_k is categorical variable,

$$p(X_k|C=i) = \frac{S_{i,k}}{S_i}$$

As an example, the proportion of X_1 in samples which belong to category $C = 1$; If X_k is continuous attribute, there is a normal distribution assumption:

$$p(X_k|C=i) = f(X_k, u_{c=i}, \sigma_{c=i}) = \frac{1}{\sqrt{2\pi}\sigma_{c=i}} exp(\frac{(x_k - u_{c=i})^2}{2\sigma_{c=i}^2})$$

$f(X_k, u_{c=i}, \sigma_{c=i})$ is the density function of the attribute X_k, $u_{c=i}$, $\sigma_{c=i}$ is the mean and the standard deviation. Hence the naive Bayes classification model is used as classifier. The following part we use Python to implement a Naïve Bayes Classifier.

Basically, the method we use 4 steps to implement Bayes Classifier in Python:

1. We input the csv file and convert it as dictionary:

```
import csv
import pprint

reader = csv.DictReader(open('data 1979.csv', 'r', encoding='utf-8'))

dict_list = []

for line in reader:
    dict_list.append(line)

pprint.pprint(dict_list)

[OrderedDict([('City', 'Los Angeles'),
              ('Race', 'White'),
              ('Gender', 'Female'),
              ('Class', 'Private'),
              ('Age', 'above 55'),
              ('Education', 'undergraduate'),
              ('Wage', 'medium')]),
 OrderedDict([('City', 'Los Angeles'),
              ('Race', 'White'),
              ('Gender', 'Male'),
              ('Class', 'Private'),
              ('Age', '16 to 25'),
              ('Education', 'undergraduate'),
              ('Wage', 'high')]),
 OrderedDict([('City', 'Los Angeles'),
              ('Race', 'Black'),
              ('Gender', 'Female'),
              ('Class', 'Private'),
              ('Age', '26 to 35'),
              ('Education', 'undergraduate'),
              ('Wage', 'medium')]),
......
```

2. Calculate the prior probability $P(C = i) = \frac{S_i}{S}$:

Compute the prior probability of a specified class (for example class of wage level).

```
#Minxuan
#Naive Bayes

#Calculate the Prob. of class:cls
def P(data,cls_val,cls_name="Wage"):
    cnt = 0.0
    for e in data:
        if e[cls_name] == cls_val:
            cnt += 1

    return cnt/len(data)
```

3. Calculate the posterior probability $p(X_k|c = i) = \frac{S_{i,k}}{S_i}$:

Compute the probability of a specified attribute given a specified class (for example race is white and city is LA).

```
#Calculate the Prob(attr|cls)
def PT(data,cls_val,attr_name,attr_val,cls_name="Wage"):
    cnt1 = 0.0
    cnt2 = 0.0
    for e in data:
        if e[cls_name] == cls_val:
            cnt1 += 1
            if e[attr_name] == attr_val:
                cnt2 += 1
    return cnt2/cnt1
```

4. Naive Bayes Algorithm:

The class vectors cls_val are transformed into 0 vectors, and compute the probability of each corresponding class.

```
#Calculate the Native Bayes
def NB(data,test,cls_0,cls_1,cls_2):
    P0 = P(data,cls_0)
    P1 = P(data,cls_1)
    P2 = P(data,cls_2)
    for key,val in test.items():
        print (key,val)
        P0 *= PT(data,cls_0,key,val)
        P1 *= PT(data,cls_1,key,val)
        P2 *= PT(data,cls_2,key,val)
    return {cls_0:P0,cls_1:P1,cls_2:P2}
```

After building the function for calculate the prior and posterior probability, the system is able to predict the class of a new sample from its features. The data set provide preliminary information to train the classifier. For example, as we input the features X_1 to X_6 of:

```
Race=White
Gender=Female
City=Los Angeles
Education=undergraduate
Class=Private
Age=26 to 35
```

into the function,

```
if __name__ == "__main__":

    #data
    data = dict_list

    #calculate
    print (NB(data,{"Race":"White","Gender":"Male","City":"Los Angeles",
                                                    "Education":"high school","Class
                                                    ":"Private","Age":"16 to 25"},"
                                                    low","medium","high")
```

The output is:

```
Race White
Gender Male
City Los Angeles
```

```
Education high school
Class Private
Age 16 to 25
{'low': 0.010301446877575974, 'medium': 0.0067023703069833225, 'high': 0
                                     .00028063538084367067}
```

With Naive Bayesian functions to calculate the posterior probability for each class, the class with the highest posterior probability is the outcome of prediction. In our case, the sample is most likely to receive a **medium-level** wage according to its characteristics.

3.2 Particle Filters

Generic Particle Filter Algorithm

1. Randomly generate a bunch of particles

Particles can have position, heading, and/or whatever other state variable you need to estimate. Each has a weight (probability) indicating how likely it matches the actual state of the system. Initialize each with the same weight.

2. Predict next state of the particles

Move the particles based on how you predict the real system is behaving.

3. Update

Update the weighting of the particles based on the measurement. Particles that closely match the measurements are weighted higher than particles which don't match the measurements very well.

4. Resample

Discard highly improbable particle and replace them with copies of the more probable particles.

5. Compute Estimate

Optionally, compute weighted mean and covariance of the set of particles to get a state estimate.

To better understand how filters work, we use Kalman Filters to fit/predict the gold price series first. We assume the state transition equation and observation equation:

$$x_t = f(x_{t-1}, u_t, w_t)$$

$$y_t = h(x_t, e_t)$$

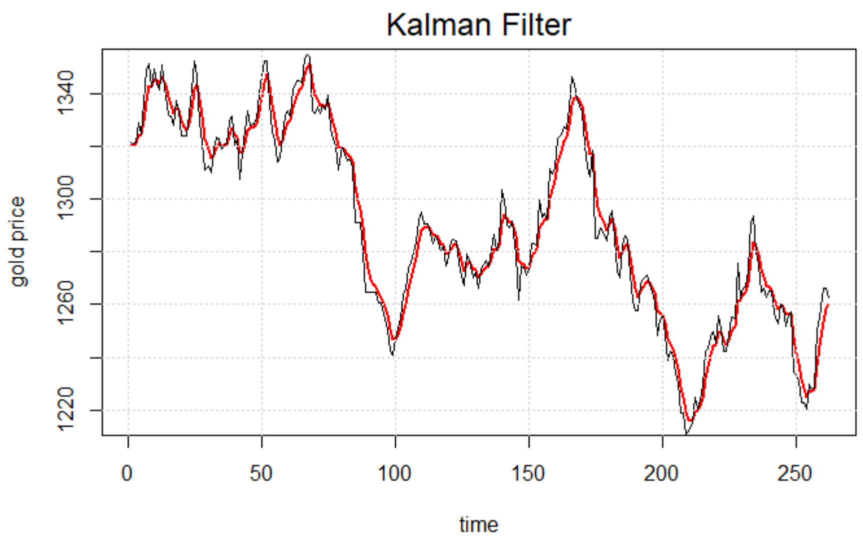
For simplify, the u_t, w_t and e_t are assumed as 0, the equations are both linear and we can define the parameters a, k as we wish. Now we can generate a simple Kalman Filter method in R to fit our gold prices data:

```
setwd("D:/Spring Quarter/412")
dat <- read.csv("1Y_GOLDUSD.csv")
data <- dat\$Value
a <- 1
k <- 0.4
x1 <- 1320 #initial Value
x2 <- 0
t <- 0
for (i in 1:length(data)) {
     x2[i] <- ((1-k)*a*x1[i])+k*data[i]
     t[i] <- i
     if (i < length(data)){x1[i+1]=x2[i]}}
plot(t, x2, type="l", ylab="gold price", xlab="time", pch=20, col='red',lwd=2)
lines(data)
grid()
title(main = list("Kalman Filter", cex=1.5, col="black", font=1))
```

Now we know how filters can fit the real data through the state and observations, if we change the value of k, which is between 0 and 1, the state and observation have different weights. However, we only use one variable so it can't be used to identify the causality between the variables in our data cue, which means we have to move into next part - The Bayesian Causal Impact Analysis[4].

This paper proposes to infer causal impact on the basis of a diffusion-regression state-space model that predicts the counterfactual market response in a synthetic control that

[4]INFERRING CAUSAL IMPACT USING BAYESIAN STRUCTURAL TIME-SERIES MODELS By Kay H. Brodersen, Fabian Gallusser, Jim Koehler, Nicolas Remy and Steven L. Scott



would have occurred had no intervention taken place. In contrast to classical difference-in-differences schemes, state-space models(which is exactly the same as the state and observation that we talked before) make it possible to incorporate empirical priors on the parameters in a **fully Bayesian treatment** using a Markov chain Monte Carlo algorithm for posterior inference. The CausalImpact R package provides an implementation of this approach.

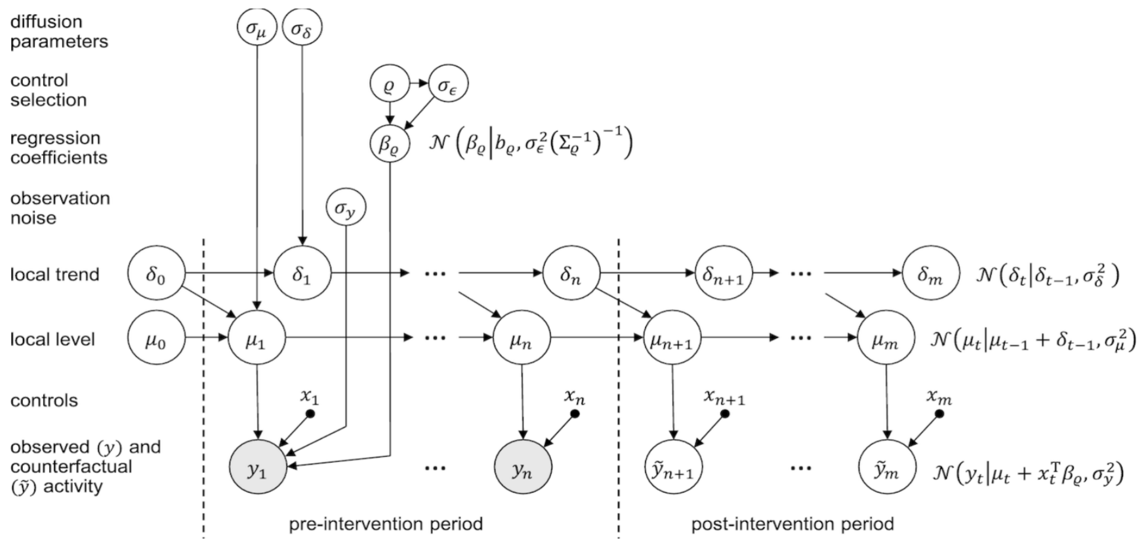
Structural time-series models are state-space models for time-series data. They can be defined in terms of a pair of equations, which is the same as we saw in Kalman Filters:

$$y_t = Z_t T \alpha_t + \varepsilon_t$$

$$\alpha_{t+1} = T_t \alpha_t \ + R_t \eta_t$$

where $\varepsilon_t \sim N(0, \sigma_t^2)$ and $\eta_t \sim N(0, Q_t)$ are independent of all other unknowns. Equation 1 is the observation equation; it links the observed data y_t to a latent d-dimensional state vector α_t. Equation 2 is the state equation; it governs the evolution of the state vector α_t through time. In the present paper, y_t is a scalar observation, Z_t is a d-dimensional output vector, T_t is a $d \times d$ transition matrix, R_t is a $d \times q$ control matrix, ε_t is a scalar observation error with noise variance σ_t, and η_t is a q-dimensional system error with a $q \times q$ state-diffusion matrix Q_t, where $q \leq d$.

Also, it indeed has a particle filter algorithm like this: To illustrate the practical utility of



our approach, we analyzed the Gold prices. In particular, we inferred the gold prices's causal effect on the Crude Oil Prices.

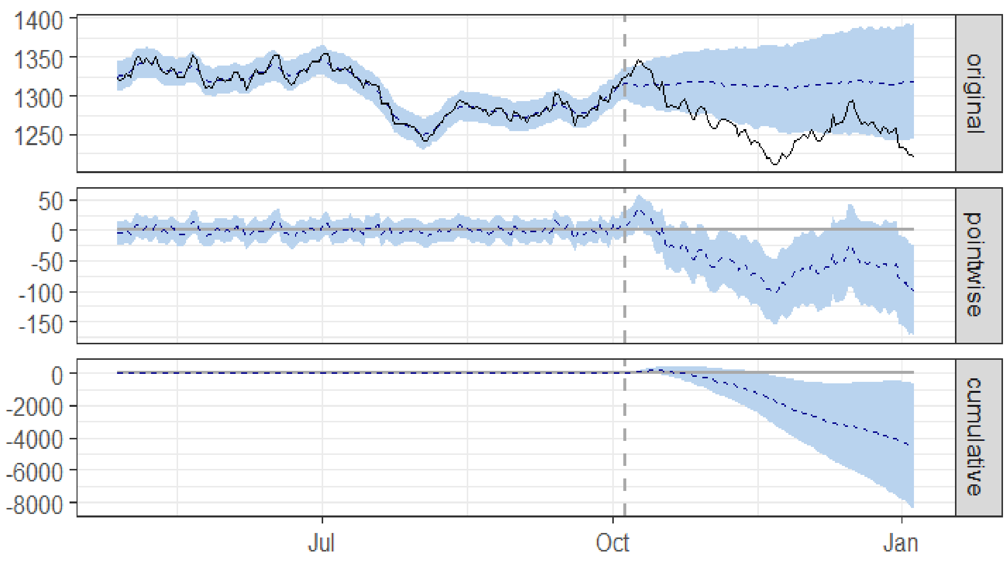
```
setwd("D:/Spring Quarter/412")
gold <- read.csv("1Y_GOLDUSD.csv")
gold <- gold\$Value[1:254]
oil <- read.csv("1Y_OILUSD.csv")
oil <- oil\$Value
time.points <- seq.Date(as.Date("2017-04-27"), by = 1, length.out = 254)
data <- zoo(cbind(gold, oil), time.points)
head(data)
              gold   oil
2017-04-27 1321.50 68.11
2017-04-28 1320.70 68.18
2017-04-29 1321.65 68.00
2017-04-30 1328.85 67.66
2017-05-01 1324.30 67.61
2017-05-02 1336.75 68.26
```

We can now specify the pre-period and the post-period in terms of time points rather than indices:

```
pre.period <- as.Date(c("2017-04-27", "2017-10-05"))
post.period <- as.Date(c("2017-10-06", "2018-01-05"))
```

As a result, the x-axis of the plot shows time points instead of indices:

```
impact <- CausalImpact(data, pre.period, post.period)
plot(impact)
```



To obtain a numerical summary of the analysis, we use:

```
summary(impact)
Posterior inference {CausalImpact}

                         Average          Cumulative
Actual                   1265             116406
Prediction (s.d.)        1315 (22)        120982 (2014)
95% CI                   [1272, 1357]     [117032, 124822]

Absolute effect (s.d.)   -50 (22)         -4577 (2014)
95% CI                   [-91, -6.8]      [-8417, -626.4]

Relative effect (s.d.)   -3.8% (1.7%)     -3.8% (1.7%)
95% CI                   [-7%, -0.52%]    [-7%, -0.52%]

Posterior tail-area probability p:   0.01333
Posterior prob. of a causal effect:  98.667%

For more details, type: summary(impact, "report")
```

During the post-intervention period, the response variable had an average value of approximate 1.27K. By contrast, in the absence of an intervention, we would have expected an average response of 1.32K. The 95% interval of this counterfactual prediction is [1.27K, 1.36K]. Subtracting this prediction from the observed response yields an estimate of the causal effect the intervention had on the response variable. This effect is -0.05K with a 95% interval of [-0.09K, -0.01K].

Summing up the individual data points during the post-intervention period (which can only sometimes be meaningfully interpreted), the response variable had an overall value of 116.41K. By contrast, had the intervention not taken place, we would have expected a sum of 120.98K. The 95% interval of this prediction is [117.03K, 124.82K].

The above results are given in terms of absolute numbers. In relative terms, the response variable showed a decrease of -4%. The 95% interval of this percentage is [-7%, -1%].

This means that the negative effect observed during the intervention period is statistically significant.

The probability of obtaining this effect by chance is very small (Bayesian one-sided tail-area probability p = 0.013). This means the causal effect can be considered statistically significant.

4 Conclusion

In this project, we utilize the Bayes theory in three algorithms. In the first module, we use a Naïve Bayes Classifier to predict the class of wage level from the attributed sample feature. We include city, race, gender, class of worker, age, education level into our current sample as the features. With our algorithm, we are able to determine the wage class of a sample by its highest posterior probability.

In the second module, we use the particle filter to conduct estimation of gold prices and analyze causal relations between gold prices and crude oil prices. The estimation line we generated from the Kalman Filter fits the real data. Also, we are able to adjust the parameter to decide the weight between historical data and observation.

From the final algorithm, we use the Bayesian Causal Impact Analysis to conclude that there is a significant causal relationship between crude oil price and gold price. Moreover, we can apply this algorithm to other time series to determine their relationship.