CS 3753 & 5163 Data Science Summer 2020

Homework 4 (100 points)

Submission:

1. submit a single python script (abc123_hw#.ipynb or abc123_hw#.py) through blackboard. All the results are outputted from your Python code.
2. You should have the instruction of running your code at the beginning of your code. It should run successfully either in the basic Python3 environment or in Jupyter Notebook.
3. Do not compress your files
4. The late submission will lose 15% points. Your code should run successfully. There is a limit of half points max if the code cannot run.
5. You can submit your homework 3 times before the deadline.

Questions

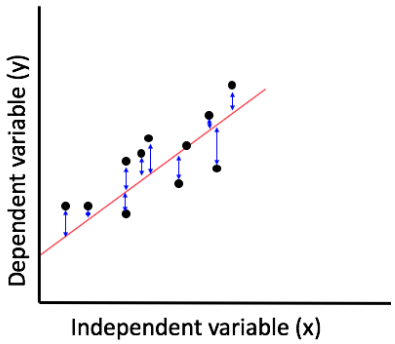
1. Linear Regression (40 points)

If the plot of n pairs of data (x , y) for an experiment appear to indicate a "linear relationship" between y and x, then the method of least squares may be used to write a linear relationship between x and y.

The least squares regression line is the line that minimizes the sum of the squares of the errors (SSE) from each data point to the line (see figure below). The least square regression line for the set of n data points is given by $y = \beta x + \alpha$, where α and β are given by

$$\beta = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\alpha = \bar{y} - \beta * \bar{x}$$



The sales of a company (in million dollars) for each year are shown in the table.

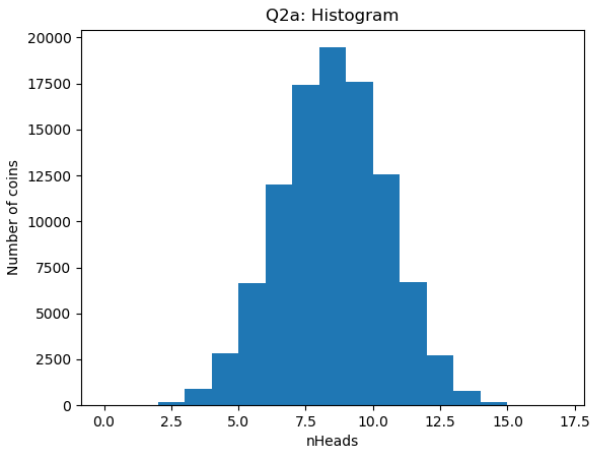
x(year)	2005	2006	2007	2008	2009
y(sales)	12	19	29	37	45

a. Write a Python code to find the least square regression line $y = \beta x + \alpha$. You can write a Python code to calculate α and β based on the equations above or use the linear regression function in the model sklearn to get the results directly). (10 pts)
b. Implement the gradient descent to calculate α and β. If you cannot get the same α and β, please explain it. (25 pts)
c. Use the regression line as a model to estimate the sales of the company in 2012. (5 pts)

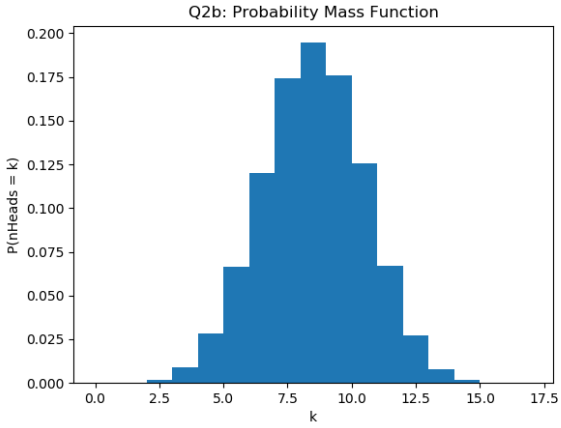
2. Statistics (60 points)

Write a program to simulate an experiment of tossing a fair coin 16 times and counting the number of heads. Repeat this experiment 10**5 times to obtain the number of heads for every 16 tosses; save the number of heads in a vector of size 10**5 (nHeads). You should be able to do this in just a few lines. (Use np.random.uniform to generate a 2d array of 10**5 x 16 random numbers between 0 and 1; a value that is greater than 0.5 is considered a "head".) Complete the following questions.

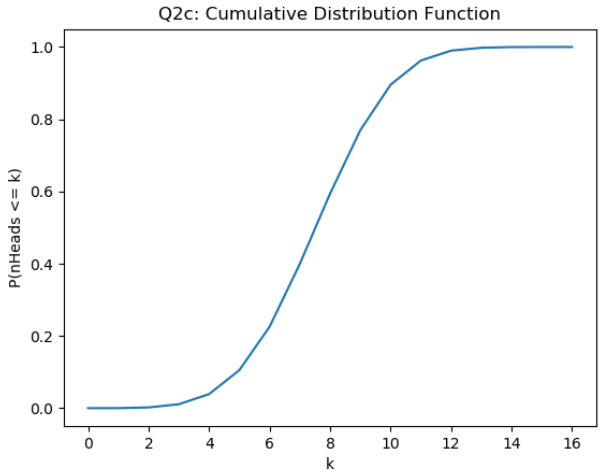
a. Plot the histogram of nHeads using plt.hist, with parameter bins = range(18). Label your plots clearly (10pts).



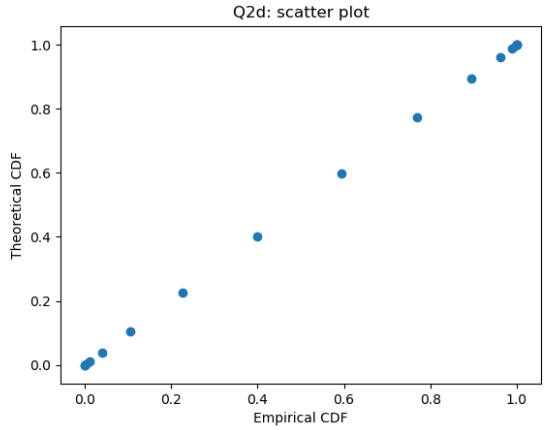
b. Plot the PMF using plt.hist with parameter bins = range(18) and normed=True. Label your plots clearly (10pts).

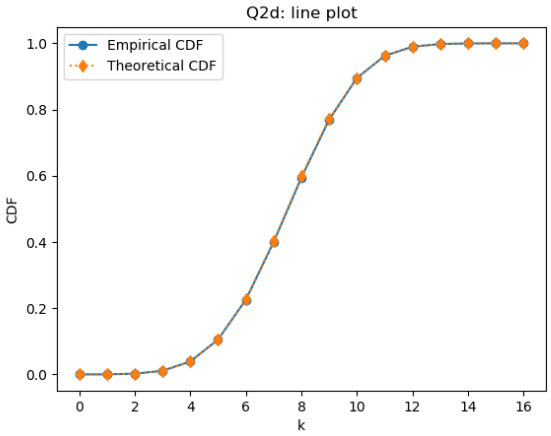


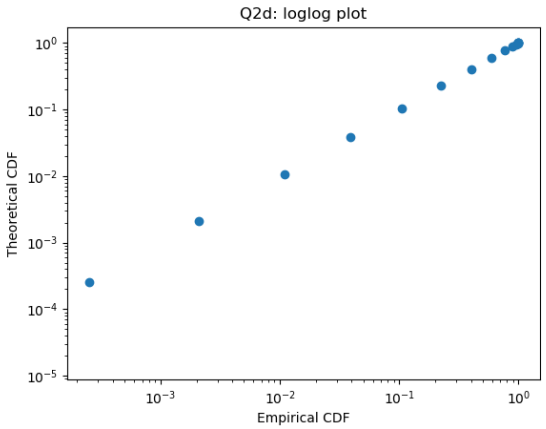
c. Calculate the probability of having NO MORE THAN k heads out of 16 tosses, where k = 0, 1, 2, … 15, 16. Plot this as a CDF. You can calculate the probabilities/counts with your own code or using values returned from 2a/2b.) Label your plots clearly (10pts).



d. Use the binomial distribution CDF (use scipy.stats.binom.cdf) to compute the probability of having NO MORE THAN k heads out of 16 tosses, where k = 0, 1, 2, …, 15, 16 and compare these probabilities with the probabilities you obtained in 2c. (Plot the probabilities you obtained from the simulation results in 2c against the probabilities from your theoretical calculation here, as **a scatter plot and line graph**. Plot in **loglog scale** to visualize small probabilities.) (30pts)


Q2d: scatter plot
Theoretical CDF
Empirical CDF
0.0
0.2
0.4
0.6
0.8
1.0


Q2d: line plot
Empirical CDF
Theoretical CDF
CDF
k
0
2
4
6
8
10
12
14
16


Q2d: loglog plot
Theoretical CDF
Empirical CDF
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}