In the name of God

University of Tehran
College of Engineering
Faculty of ECE

Intelligent Systems

HW5

Reza Jahani

810198377

Fall 1401

Table of Contents

Statistical Problems Implementation ***3***

- ***Section 1*** ***3***
- ***Section 2*** ***5***

Analytical Statistical Problems ***7***

- ***Section 1*** ***7***
- ***Section 2*** ***8***

Naïve Bayes ***9***

Additional Section ***12***

Question 1) Statistical Problems Implementation

The secretary problem demonstrates a scenario involving optimal stopping theory that is studied extensively in the fields of applied probability, statistics, and decision theory. It is also known as the marriage problem, the sultan's dowry problem, the fussy suitor problem, the googol game, and the best choice problem.

Section 1)

Part 1)

In this part we aim to employ an algorithm which finds the best combination of features to choose the best candidate by observing the first ***k*** candidates and rejecting them and then using that figured feature to choose the very first candidate who satisfies the following criteria.
Employing this algorithm, the probability of correct decision is calculated in each ***k*** and the plot can be observed in figure 1.1.1.

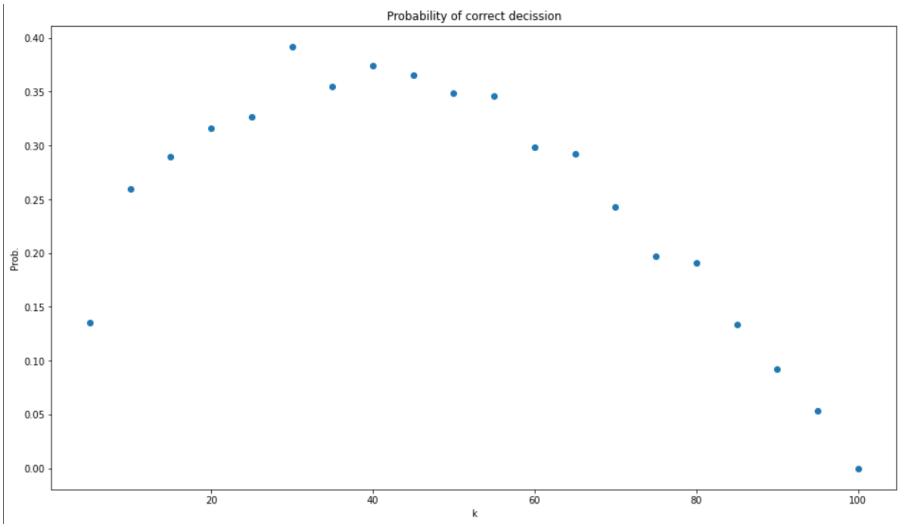

Figure 1.1.1: Success Prob. Per ***k***

As observed in figure 1.1.2, the best value for ***k*** is illustrated.

```
best_k = K[np.argmax(success_prob)]
print(f"Best K = {best_k}")

Best K = 30
```

Figure 1.1.2: Best Value of ***k***

Part 2)

In this part we keep the value of $k = \frac{n}{e}$ and iterate twice. First on n values from 3 to 100 and secondly on iterations of experiments. Employing the algorithm, the best candidate is decided again and for each n value, the probability of correct decision is computed. In figure 1.1.3, the plot of this probability concept per n is scattered.

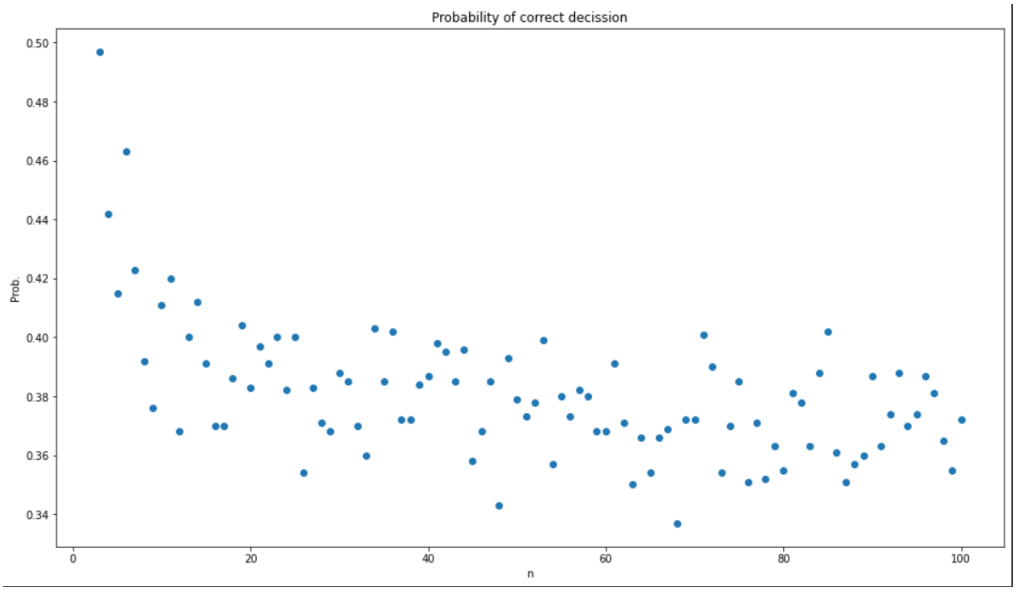

Figure 1.1.3: Correct decision probability per number of candidates

Part 3) Analytical Calculation of Correct Decision Probability

$$For\ the\ strategy\ to\ win\ in\ each\ iteration:\ \ p = \frac{k}{n}\frac{1}{N}$$

$$P_R = \frac{1}{N}\left(\frac{k}{k} + \frac{k}{k+1} + \frac{k}{k+2} + \cdots \frac{k}{N-1}\right) = \frac{k}{N}\sum_{n=r}^{N-1}\frac{1}{n}$$

$$P_R = \lim_{N\to\infty} \frac{k}{N}\sum_{n=r}^{N-1}\frac{1}{n}.\frac{N}{N} = x\int_x^1 \frac{1}{t}dt = -xln(x)$$

$$\frac{d}{dx}P_R = 0 \to x = \frac{1}{e}$$

$$P_R = \frac{1}{e}$$

Section 2)

In this section we have a distribution. For s=100 times we take n=1000 random samples from the following distribution and in each time, the mean of the sample set is calculated. The histogram of this parameter is observed in figure 1.2.1

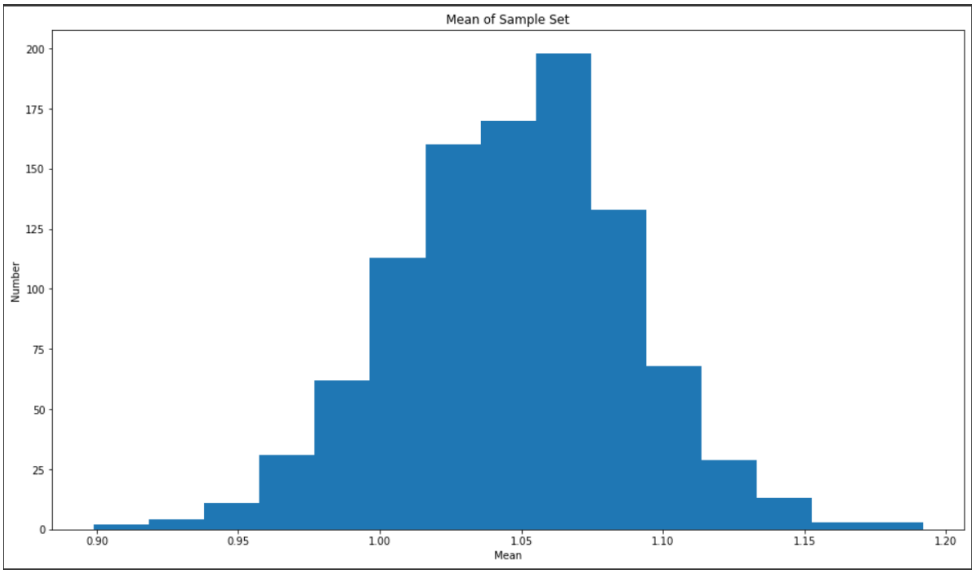

Figure 1.2.1: Histogram Plot of Sample set mean

After reading the csv file of the wine dataset, 12'th column is separated and its distribution is plotted and presented in figure 1.2.3.

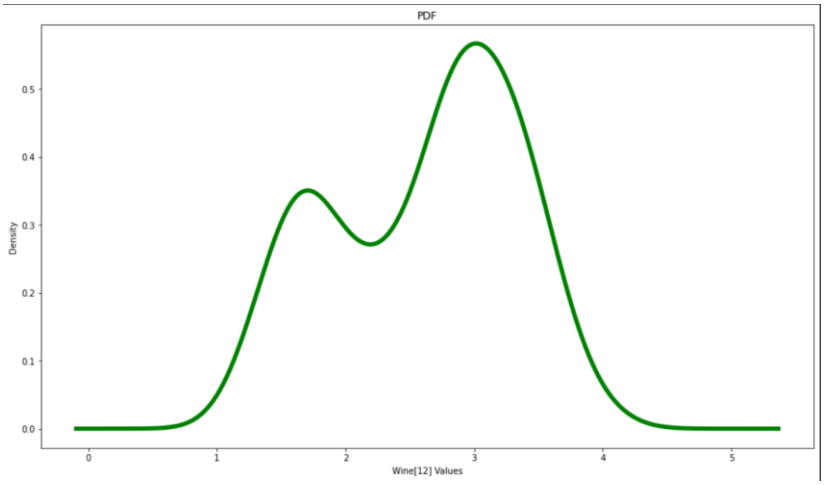

Figure 1.2.3: Distribution of Wine [12]

Employing the same algorithm conducted in the beginning of this question, samples of this population (12'th column of wine) is taken with different sizes and the mean is calculated then the histogram is plotted. This histogram can be observed in figure 1.2.4.

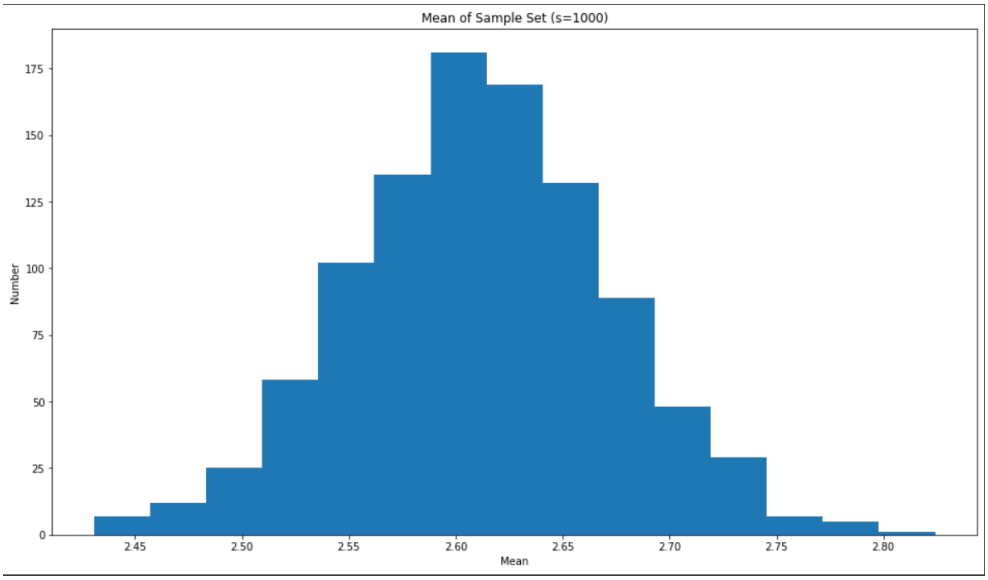

Figure 1.2.4

As it can be seen above, the plot is so close to the figure of normal distribution. It is due to the CLT concept that summation of N random variables tends to normal distribution when N gets higher and higher.

Comparison with analytical equations of CLT:

$$According\ to\ CLT:$$

$$Z = \frac{1}{n}\sum X_i \rightarrow N(m, \sigma^2)$$

$$m = m_X\ , \sigma^2 = \frac{1}{\sqrt{n}}\sigma_X^2$$

As stated above, equations apply and in figure 1.2.5, we can confirm this fact. Since we iterated for 1000 times approximately, the mean value is quite the same and the value for standard deviation (which is square root of σ^2, is approximately the value obtained by the simulation.

```
Wine[12] Mean: 2.612
Samplesets Mean: 2.616
Wine[12] Standard Deviation: 0.708
Samplesets Standard Deviation: 0.074
```

Figure 1.2.5: CLT Theorem Confirmation

Question 2) Analytical Statistical Problems

Section 1)

We already know that employing Bayesian Classifier using posterior probabilities and prior probabilities we would have:

$$C_{MAP} = argmax\,\{P(c_i) f_X(x_1, x_2, \dots, x_n|c_i)\}$$

In Naïve Bayes approach we consider the distribution PDF as a multiplication of individual PDF for each feature. In this case we would have:

$$f_X(x_1, x_2, \dots, x_n|c_i) = f_{X_1}(x_1|c_i) f_{X_2}(x_2|c_i) \dots f_{X_n}(x_n|c_i) = \prod_{m=1}^{n} f_{X_m}(x_m|c_i) \quad [eq.\,2.1]$$

$$C_{NB} = argmax\left\{P(c_i) \prod_{m=1}^{n} f_{X_m}(x_m|c_i)\right\}$$

According to the hypothesis of the question, after applying a kernel and mapping the data to another space, features would be uncorrelated and perpendicular.

$$for\ i \neq j\ ;\ r_{X_i X_j} = E\{X_i X_j\} = 0$$
$$for\ i \neq j\ ;\ c_{X_i X_j} = E\{X_i X_j\} - E\{X_i\} E\{X_j\} = 0$$

Above given data from the question does not necessarily assures us from independency of features. This means that features are not necessarily independent and equation 2.1 is not necessarily true unless the features have normal distribution.

In conclusion, Ahmad's illustration is not correct, and his solution in real world still face problems and errors.

Under only one circumstance, his illustration would be true and it is the statement that the features in mapped space should be normal. In this case they would be independent according to their uncorrelated and perpendicular fact.

Section 2)

Since the data is from a normal distribution:

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-m_x)^2}{2\sigma^2}\right)$$

$$\log f_X = -\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\sigma^2) - \frac{1}{2}\frac{(x-m_x)^2}{\sigma^2}$$

$$L(x_1, \dots, x_n; m_x, \sigma^2) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2}\sum_{i=1}^{n}\frac{(x_i-m_x)^2}{\sigma^2}$$

$$\frac{\partial}{\partial m_x} L = 0 \rightarrow \frac{1}{\sigma^2}\sum_{i=1}^{n} x_i - n m_x = 0$$

$$\frac{\partial}{\partial \sigma^2} L = 0 \rightarrow -\frac{n}{2\sigma^2} + \frac{1}{s\sigma^4}\sum_{i=1}^{n}(x_i-m_x)^2 = 0$$

$$m_x = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-m_x)^2$$

Question 3) Naïve Bayes

Section 1) Feature Extraction & Classifier Design

In this section we aim to design the classifier by 4 features that is going to be extracted for each class. The features which are employed to design the classifier are explained below.

- Number of # in the picture
- Number of + in the picture
- Height of the image
- Width of the image

After extraction of the features, distribution of each feature conditioned to each class should be figured. For this purpose, normal distribution is employed to be fit to each feature. As we already know, the distribution given to a normal distribution is based on mean and standard deviation. So these parameters are calculated for each of the features conditioned to each class.

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-m)^2}{2\sigma^2}\right)$$

For each feature, mean and standard deviation are calculated. Furthermore, short explanations of the algorithm of feature extraction and model design is presented.

```
[127] # Feature 1: Number of # in the image
      # For all dataset
      def feature_1_extractor(data,N):
          temp = np.reshape(data,(N,DATA_WIDTH*DATA_HEIGHT))
          return np.sum(temp==1,axis=1)

[128] # Feature 1: Number of + in the image
      # For all dataset
      def feature_2_extractor(data,N):
          temp = np.reshape(data,(N,DATA_WIDTH*DATA_HEIGHT))
          return np.sum(temp==2,axis=1)
```

Figure 3.1: Feature 1 & 2

As it can be observed for figuring the number of #s and +s in the image is calculated by summation of times that the 2D array contains # and +, since # is mapped to integer 1 and + is mapped to integer 2. The algorithm employed for finding height and width of the image is observed in figure 3.2. For this aim, the last row(column) from above containing all zero values and first row(column) from below containing all zero values are found and the difference between these two values are considered as height(width).

```python
# Feature 3: Height of image
# For one image
def feature_3_extractor(data):
    h1 = 0
    h2 = 27
    not_empty_rows = []
    empty_rows = []
    for i in range(DATA_HEIGHT):
        row = data[i]
        if np.sum(row) == 0:
            empty_rows.append(i)
        else:
            not_empty_rows.append(i)

    for i in range(DATA_HEIGHT):
        if i in empty_rows and (i+1) in not_empty_rows:
            h1 = i
        if i in empty_rows and (i-1) in not_empty_rows and (i+1) in empty_rows:
            h2 = i

    return h2-h1
```

```python
# Feature 4:Width of image
# For one image
def feature_4_extractor(data):
    w1 = 0
    w2 = 27
    data_ = np.transpose(data)
    not_empty_rows = []
    empty_rows = []
    for i in range(DATA_WIDTH):
        row = data_[i]
        if np.sum(row) == 0:
            empty_rows.append(i)
        else:
            not_empty_rows.append(i)

    for i in range(DATA_HEIGHT):
        if i in empty_rows and (i+1) in not_empty_rows:
            w1 = i
        if i in empty_rows and (i-1) in not_empty_rows and (i+1) in empty_rows:
            w2 = i

    return w2-w1
```

Figure 3.2: Feature 3 & 4

After extraction each feature conditioned to each class, mean and standard deviation is computed to fit a normal distribution to each feature.

Section 2) Model Evaluation on Validation Dataset

Here is a short explanation about the prediction and model process. Naïve Bayes model functions this way that for each data, those features are extracted and then assigns independent probabilities of possession to a certain class based on the observation according to the fit distribution of a feature. Considering these probabilities independent, overall probability of possession to a class is computed by multiplication of these probabilities. After computing probabilities of possession to each class, the class which has the highest probability is considered as the final label. Mathematical and theoretical illustrations are presented below.

$$P(c|x) = \frac{P(X|c)P(c)}{P(X)}$$

$$C_{MAP} = argmax\ P(c|X) = argmax\ \frac{P(X|c)P(c)}{P(X)} = argmax\ P(X|c)P(c)$$

Since P(c) is similar for all classes, this term would not be effective in our decision, so we won't consider it. X is the observation vector(Features).

$$C_{MAP} = argmax\ P(X|c) = argmax\ P(x_1, x_2, \dots, x_n|c)$$

In Naïve Bayes model all features are considered independent so the distribution can be derived as below.

$$P(x_1, x_2, \dots, x_n|c) = P(x_1|c)P(x_2|c)P(x_3|c) \dots P(x_n|c)$$

In conclusion, we have:

$$C_{NB} = argmax \prod_{k=1}^{n} P(x_k|c)$$

As already stated in previous section, these distributions are considered normal.

```
[176] def naive_bayes_model(X_test,f1,f2,f3,f4):
          x1 = feature_1_extractor(X_test,1)
          x2 = feature_2_extractor(X_test,1)
          x3 = feature_3_extractor(X_test)
          x4 = feature_4_extractor(X_test)
          P = []
          for i in range(10):
              p1 = normal(x1,f1[i][0],f1[i][1])
              p2 = normal(x2,f2[i][0],f2[i][1])
              p3 = normal(x3,f3[i][0],f3[i][1])
              p4 = normal(x4,f4[i][0],f4[i][1])
              p = p1 * p2 * p3 * p4
              P.append(p)
          predicted_label = np.argmax(np.array(P))
          return predicted_label
```

Figure 3.3: Naïve Bayes Model Process

Using the validation dataset, the model is evaluated and the result is observed below in figure 3.4. Model's performance is adequate enough since only 4 features are employed.

```
Naive Bayes Model Accuracy:
27.4%
```

Figure 3.4: Model's Accuracy

Additional Section

- All of the implementations for this HW are in the google colab file named ***HW5_810198377.ipynb***.
- All questions and their sections and parts are separated with the related headings and titles.