Machine Learning HW #2 answer

1.

p(patient is infected) = 0.01%

FP = p(reagent is negative | patient is infected) = 0.1%

FN = p(reagent is positive | patient is not infected)= 0.1%

p(patient is infected | reagent is positive)

$$= \frac{\text{p(patient is infected)p(reagent is positive | patient is infected)}}{\text{p(reagent is positive)}}$$

p(reagent is positive)

= p(patient is infected)p(reagent is positive | patient is infected)

+ p(patient is not infected)p(reagent is positive | patient is not infected)

= 0.01%*99.9% + 99.99%*0.1%

p(patient is infected | reagent is positive)

$$= \frac{0.01\% * 99.9\%}{0.01\% * 99.9\% + 99.99\% * 0.1\%} = 0.0908$$

2.

p(x) = p(B) p(c) = 6/12 4/12 = 1/6

p(x|+) = p(B|+) p(c|+) = 3/7 2/7 = 6/49

p(+|x) = p(+) p(x|+) / p(x) = 7/12 6/49 / 1/6 = 3/7

p(x|-) = p(B|-) p(c|-) = 3/5 2/5 = 6/25

p(-|x) = p(-) p(x|-) / p(x) = 5/12 6/25 / 1/6 = 3/5

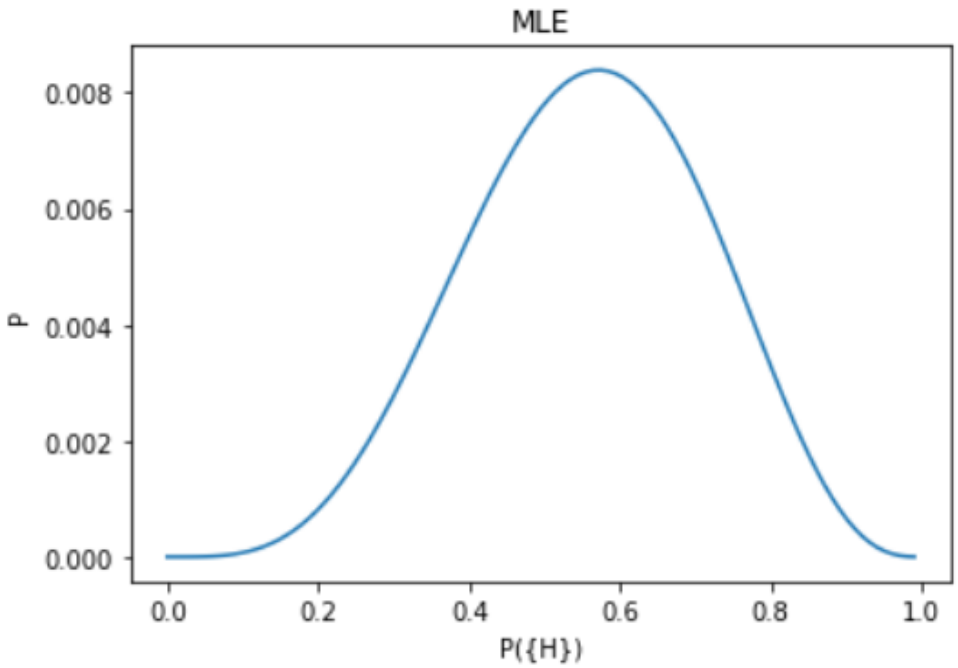
p(-|x) is greater than p(+|x), so the blue circle will be classified in -

3.

In plot (c) cannot detect how y change when x gets larger, there is no linear relationship between x and y. (c) has the smallest correlation coefficient. Otherwise, in plot (a) when x gets bigger, y gets larger either. There is linear relationship between x and y. (a) has the largest correlation coefficient. Negative correlation coefficient is when x gets larger, y gets smaller. But no plot is negative correlation coefficient.

4.

```python
from matplotlib import pyplot as plt
import math

def bernoulli(theta):
        return theta * (1-theta) * (1-theta) * (1-theta) * theta * theta * theta

X, Y = [], []
H, T = 4, 3
for idx in range(101):
        x = 1/101 * idx

        X.append(x)
        Y.append(bernoulli(x))

plt.title('MLE')
plt.xlabel('P({H})')
plt.ylabel('P')
plt.plot(X, Y)
plt.show()

print(f'θ= {X[Y.index(max(Y))]} has max P= {max(Y)}')
```



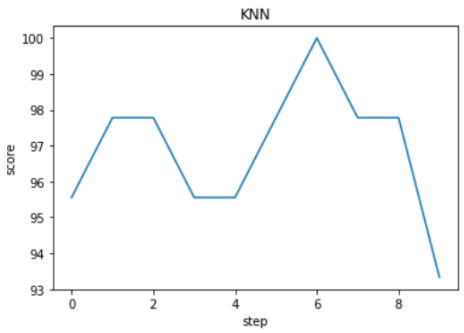
θ= 0.5742574257425742 has max P= 0.008392043464510224

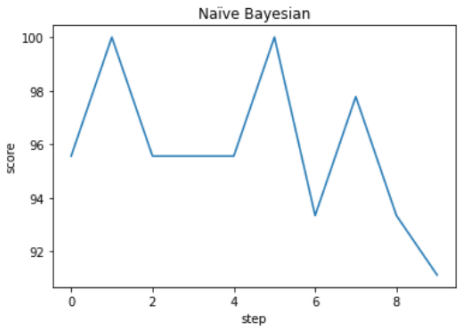
5.

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import GaussianNB

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

iris_dataset = datasets.load_iris()

df_X = iris_dataset.data[:,:4]
df_y = iris_dataset.target
```

```
KNN_scores = []
GaussianNB_scores = []
```

```
for i in range(10):
        X_train, X_test, y_train, y_test = train_test_split(df_X, df_y, test_size=0.3)

        kNN_classifier = KNeighborsClassifier(n_neighbors=7)
        kNN_classifier.fit(X_train, y_train)
        KNN_scores.append(round(accuracy_score(kNN_classifier.predict(X_test), y_test) * 100, 3))

        GaussianNB_classifier = GaussianNB()
        GaussianNB_classifier.fit(X_train, y_train)
        GaussianNB_scores.append(round(accuracy_score(GaussianNB_classifier.predict(X_test), y_test) * 100, 3))

plt.plot(KNN_scores)
plt.title('KNN')
plt.xlabel('step')
plt.ylabel('score')
plt.show()

plt.plot(GaussianNB_scores)
plt.title('Naïve Bayesian')
plt.xlabel('step')
plt.ylabel('score')
plt.show()

print(f"KNN Score = ", KNN_scores)
print(f"KNN Score average: {np.mean(KNN_scores)}")
print(f"Naïve Bayesian classifier Score = ", GaussianNB_scores)
print(f"Naïve Bayesian classifier Score average: {np.mean(GaussianNB_scores)}")
```






```
KNN Score =  [95.556, 97.778, 97.778, 95.556, 95.556, 97.778, 100.0, 97.778, 97.778, 93.333]
KNN Score average: 96.8891
Naïve Bayesian classifier Score =  [95.556, 100.0, 95.556, 95.556, 95.556, 100.0, 93.333, 97.778, 93.333, 91.111]
Naïve Bayesian classifier Score average: 95.7779
```