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)

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

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(s) = 7/13 8/13 = 56/169

p(x|+) = p(B|+) p(s|+) = 3/7 5/7 = 15/49

p(+|x) = p(+) p(x|+) / p(x) = 7/13 15/49 / 56/169 = 195/392

p(x|-) = p(B|-) p(s|-) = 4/6 3/6 = 1/3

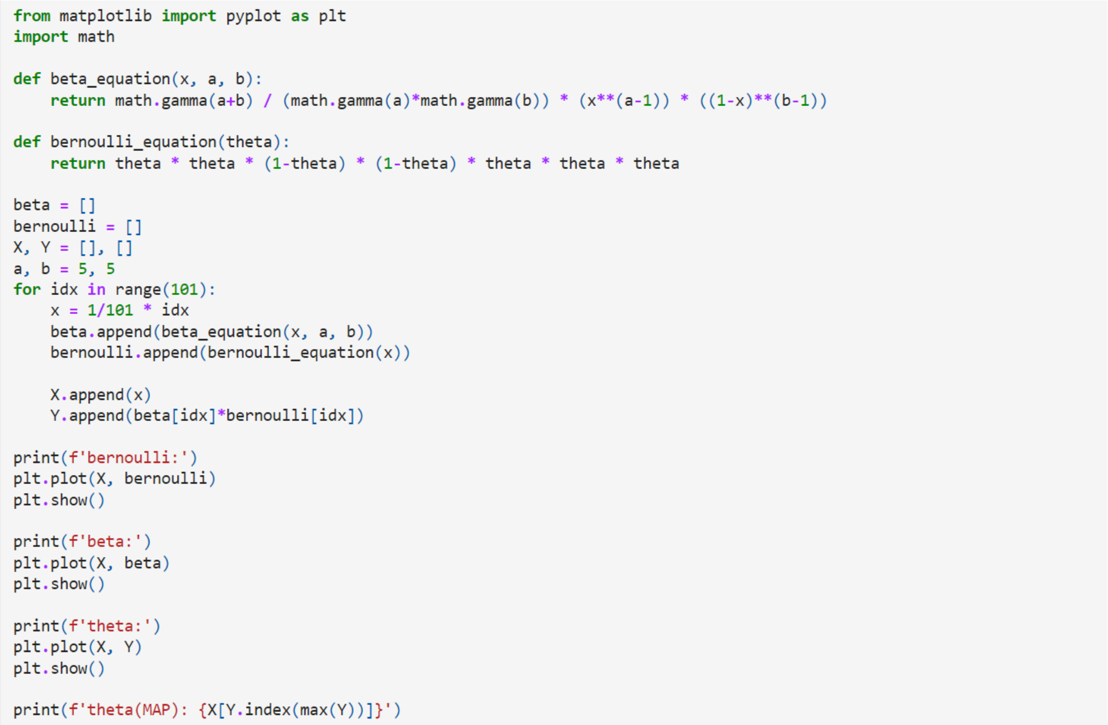
p(-|x) = p(-) p(x|-) / p(x) = 6/13 1/3 / 56/169 = 13/28

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

3.

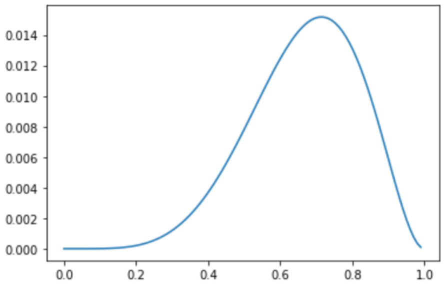
In plot (c) cannot detect how y change when x gets larger, there is no linear relationship between x and y. (c) has the smallest correlation coefficient. Otherwise, in plot (a) when x gets bigger, y gets larger either. There is linear relationship between x and y. (a) has the largest correlation coefficient.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

4.

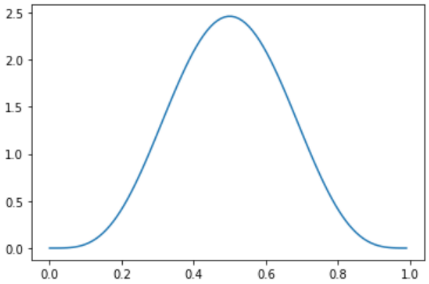


```
from matplotlib import pyplot as plt
import math

def beta_equation(x, a, b):
    return math.gamma(a+b) / (math.gamma(a)*math.gamma(b)) * (x**(a-1)) * ((1-x)**(b-1))

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

beta = []
bernoulli = []
X, Y = [], []
a, b = 5, 5
for idx in range(101):
    x = 1/101 * idx
    beta.append(beta_equation(x, a, b))
    bernoulli.append(bernoulli_equation(x))

    X.append(x)
    Y.append(beta[idx]*bernoulli[idx])

print(f'bernoulli:')
plt.plot(X, bernoulli)
plt.show()

print(f'beta:')
plt.plot(X, beta)
plt.show()

print(f'theta:')
plt.plot(X, Y)
plt.show()

print(f'theta(MAP): {X[Y.index(max(Y))]}')
```

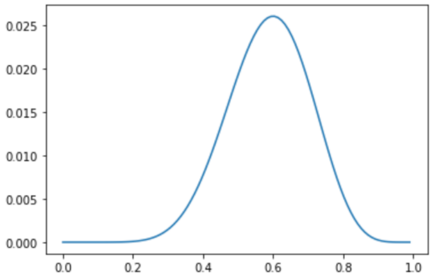
```
bernoulli:
```



```
beta:
```

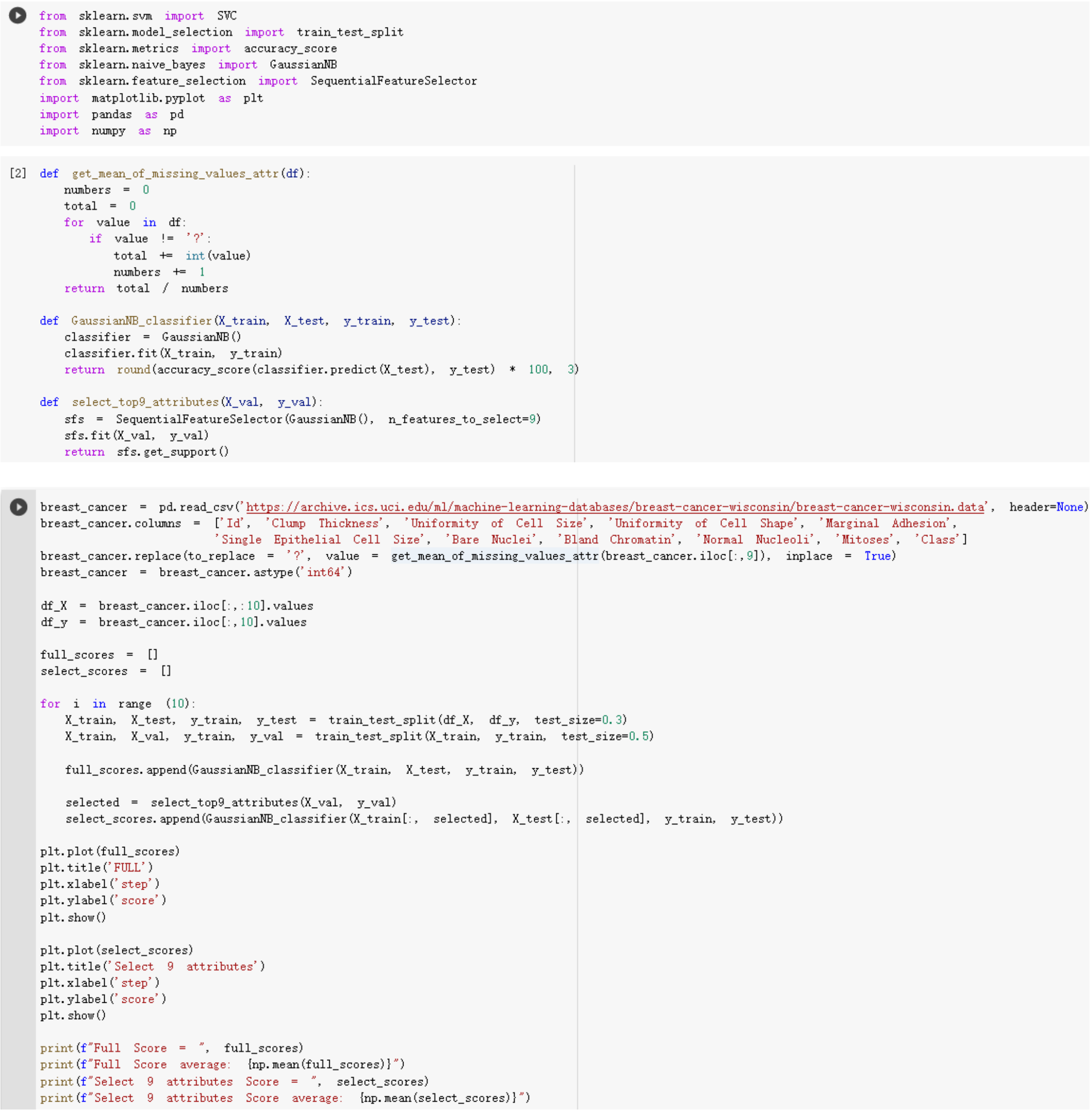


```
theta:
```



```
theta(MAP): 0.6039603960396039
```

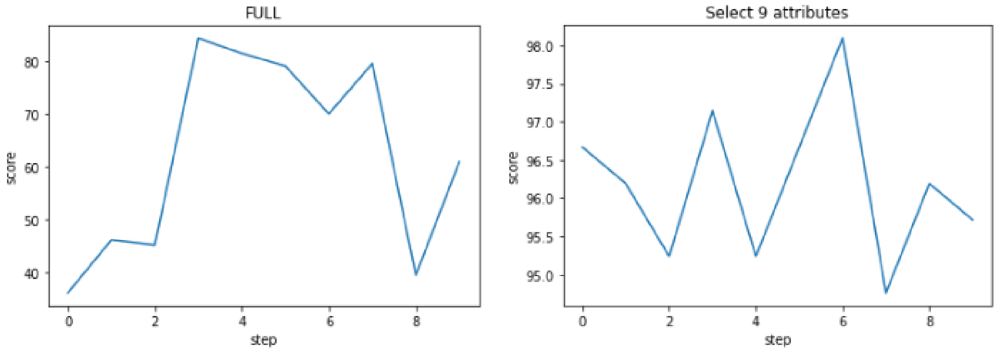
5.



```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import GaussianNB
from sklearn.feature_selection import SequentialFeatureSelector
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

```
[2] def get_mean_of_missing_values_attr(df):
        numbers = 0
        total = 0
        for value in df:
            if value != '?':
                total += int(value)
                numbers += 1
        return total / numbers

    def GaussianNB_classifier(X_train, X_test, y_train, y_test):
        classifier = GaussianNB()
        classifier.fit(X_train, y_train)
        return round(accuracy_score(classifier.predict(X_test), y_test) * 100, 3)

    def select_top9_attributes(X_val, y_val):
        sfs = SequentialFeatureSelector(GaussianNB(), n_features_to_select=9)
        sfs.fit(X_val, y_val)
        return sfs.get_support()
```

```
breast_cancer = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data', header=None)
breast_cancer.columns = ['Id', 'Clump Thickness', 'Uniformity of Cell Size', 'Uniformity of Cell Shape', 'Marginal Adhesion',
                         'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin', 'Normal Nucleoli', 'Mitoses', 'Class']
breast_cancer.replace(to_replace = '?', value = get_mean_of_missing_values_attr(breast_cancer.iloc[:,9]), inplace = True)
breast_cancer = breast_cancer.astype('int64')

df_X = breast_cancer.iloc[:,:10].values
df_y = breast_cancer.iloc[:,10].values

full_scores = []
select_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)
    X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.5)

    full_scores.append(GaussianNB_classifier(X_train, X_test, y_train, y_test))

    selected = select_top9_attributes(X_val, y_val)
    select_scores.append(GaussianNB_classifier(X_train[:, selected], X_test[:, selected], y_train, y_test))

plt.plot(full_scores)
plt.title('FULL')
plt.xlabel('step')
plt.ylabel('score')
plt.show()

plt.plot(select_scores)
plt.title('Select 9 attributes')
plt.xlabel('step')
plt.ylabel('score')
plt.show()

print(f"Full Score = ", full_scores)
print(f"Full Score average: {np.mean(full_scores)}")
print(f"Select 9 attributes Score = ", select_scores)
print(f"Select 9 attributes Score average: {np.mean(select_scores)}")
```



```
Full Score =  [36.19, 46.19, 45.238, 84.286, 81.429, 79.048, 70.0, 79.524, 39.524, 60.952]
Full Score average: 62.238099999999996
Select 9 attributes Score =  [96.667, 96.19, 95.238, 97.143, 95.238, 96.667, 98.095, 94.762, 96.19, 95.714]
Select 9 attributes Score average: 96.1904
```