EXPERIMENT – 16

PROGRAM:

```
import math


def knn_predict(X_train, y_train,
test, k=3):
    distances = []
    for i in range(len(X_train)):
        dist = sum((a - b) ** 2 for a, b in
zip(X_train[i], test))
        distances.append((dist,
y_train[i]))

    distances.sort()
    neighbors = [label for _, label in
distances[:k]]
    from collections import Counter
    return
Counter(neighbors).most_common(
1)[0][0]


def naive_bayes_predict(X_train,
y_train, test):
    classes = list(set(y_train))
    class_probs = {}

    for cls in classes:
        class_data = [X_train[i] for i in
range(len(X_train)) if y_train[i] ==
cls]
        prior = len(class_data) /
len(X_train)
        likelihood = 1.0
        for i in range(len(test)):
            feature = test[i]
            similar = sum(1 for x in
class_data if abs(x[i] - feature) <
0.5)
            likelihood *= (similar + 1) /
(len(class_data) + 2)

        class_probs[cls] = prior *
likelihood

    return max(class_probs,
key=class_probs.get)

print("Enter number of training
samples:")
n = int(input())

print("Enter number of features:")
m = int(input())

print("Enter training data (features
and class):")
X_train = []
y_train = []

for _ in range(n):
    data = input().split()
    X_train.append(list(map(float,
data[:-1])))
    y_train.append(data[-1])
```

```
print("Enter test sample features:")

test = list(map(float, input().split()))


knn_result = knn_predict(X_train,
y_train, test)

nb_result =
naive_bayes_predict(X_train,
y_train, test)


print("KNN Prediction:", knn_result)

print("Naive Bayes Prediction:",
nb_result)
```