1. **Gaussian Naive Bayes**

   Fill in the blank lines in the code below to implement Gaussian Naive Bayes. You can find the text format of this code in the homework folder.

   See this link for more information. Also see the sklearn GaussianNB source code. You may find it helpful.

```python
import numpy as np

class GaussianNB:

    def __init__(self, priors=None, var_smoothing=1e-9):
        self.priors = priors
        self.var_smoothing = var_smoothing

    def logprior(self, class_ind):
        return np.log(self.class_priors_[class_ind])

    def loglikelihood(self, Xi, class_ind):
        # mu: mean, var: variance, Xi: sample (a row of X)
        # Get the class mean
        mu =
        # Get the class variance
        var =
        # Write the Gaussian Likelihood expression
        GaussLikelihood =
        # Take the log of GaussLikelihood
        logGaussLikelihood =
        # Return loglikelihood of the sample. Now you will use the "naive"
        # part of the naive bayes.
        return

    def posterior(self, Xi, class_ind):
        logprior = self.logprior(class_ind)
        loglikelihood = self.loglikelihood(Xi, class_ind)
        # Return posterior
        return

    def fit(self, X, y):
        # Number of samples, number of features
        n_samples, n_features =
        # Get the unique classes
        self.classes_ =
        # Number of classes
        n_classes =

        # Initialize attributes for each class
        # Feature means for each class, shape (n_classes, n_features)
        self.theta_ =
        # Feature variances for each class shape (n_classes, n_features)
        self.var_ =
        # Class priors shape (n_classes,)
        self.class_priors_ =

        # Calculate class means, variances and priors
        for c_ind, c_id in enumerate(self.classes_):
            # Get the samples that belong to class c_id
            X_class =
            # Mean of the each feature that belongs to class c_id
            self.theta_[c_ind, :] = np.mean(X_class, axis=0)
            # Calculate the variance of each feature that belongs to c_id
            self.var_[c_ind, :] =
            # Calculate the priors for each class
            self.class_priors_[c_ind] =

```

```
59      def predict(self, X):
60          y_pred = []
61          for Xi in X: # Calculate posteriors for each sample
62              posteriors = []     # For saving posterior values for each class
63              # Calculate posterior probability for each class
64              for class_ind in self.classes_:
65                  # Calculate posterior
66                  sample_posterior =
67                  # Append the posterior value of this class to posteriors
68                  posteriors.append(sample_posterior)
69              # Get the class that has the highest posterior prob. and
70              # append the prediction for this sample to y_pred
71
72          # Return predictions for all samples
73          return
```