Home Assignment:

The objective of the provided code is to classify students' passing status using a Naïve Bayes algorithm based on their gender, age, and math score. This code demonstrates the application of supervised learning to train, test, and predict student performance, aiding in identifying those who may need further academic support or intervention.

- Gender: Represents the student's gender, categorized as 'Female' or 'Male'.
- Age: Indicates the age of the student in years.
- Math Score: The student's score in mathematics (0-100 scale), reflecting math proficiency.
- Passing Status: Indicates whether the student has passed ('Yes') or failed ('No'); this is the target variable for prediction.

```
import pandas as pd
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split

# Create the dataset
data = {
    'Gender': ['Female', 'Male', 'Female', 'Male', 'Female',
               'Male', 'Female', 'Male', 'Female', 'Male'],
    'Age': [17, 16, 16, 17, 18, 18, 17, 16, 17, 18],
    'Math Score': [75, 50, 85, 60, 90, 40, 55, 70, 80, 65],
    'Passing Status': ['Yes', 'No', 'Yes', 'No', 'Yes',
                       'No', 'No', 'Yes', 'Yes', 'No']
}

# Create a DataFrame
df = pd.DataFrame(data)

# Encode categorical variables
le_gender = preprocessing.LabelEncoder()
le_status = preprocessing.LabelEncoder()

df['Gender_encoded'] = le_gender.fit_transform(df['Gender'])
df['Passing_Status_encoded'] = le_status.fit_transform(df['Passing Status'])

# Define features and target variable
features = df[['Gender_encoded', 'Age', 'Math Score']]
target = df['Passing_Status_encoded']

# Split the data into training and testing sets
features_train, features_test, target_train, target_test = train_test_split(
    features, target, test_size=0.5, random_state=42
)
```

```python
# Instantiate and train the Naive Bayes model
model = GaussianNB()
model.fit(features_train, target_train)

# Make predictions on the test set
predicted = model.predict(features_test)

# Calculate the confusion matrix and accuracy
conf_mat = confusion_matrix(target_test, predicted)
accuracy = accuracy_score(target_test, predicted)

# Output the results
print("Predictions:", le_status.inverse_transform(predicted))
print("Actual Labels:", le_status.inverse_transform(target_test))
print("Confusion Matrix:\n", conf_mat)
print("Accuracy:", accuracy)

# Calculate True Positives, False Positives, True Negatives, False Negatives
TP = conf_mat[1, 1]  # True Positives
FP = conf_mat[0, 1]  # False Positives
TN = conf_mat[0, 0]  # True Negatives
FN = conf_mat[1, 0]  # False Negatives

print("True Positives (TP):", TP)
print("False Positives (FP):", FP)
print("True Negatives (TN):", TN)
print("False Negatives (FN):", FN)

# Calculate and print the accuracy manually
calculated_accuracy = (TP + TN) / (TP + TN + FP + FN)
print("Calculated Accuracy:", calculated_accuracy)
```

```
Predictions: ['Yes' 'No' 'No' 'Yes' 'No']
Actual Labels: ['Yes' 'No' 'No' 'Yes' 'Yes']
Confusion Matrix:
 [[2 0]
 [1 2]]
Accuracy: 0.8
True Positives (TP): 2
False Positives (FP): 0
True Negatives (TN): 2
False Negatives (FN): 1
Calculated Accuracy: 0.8
```