```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

def load_data():
    # Synthetic data for Bank Loan Prediction
    data = {
        'Age': [25, 35, 45, 20, 30, 50, 40, 60, 48, 33],
        'Job': ['Yes', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'Yes',
'No'],
        'Income': [50000, 60000, 30000, 25000, 45000, 80000, 40000, 30000,
70000, 42000],
        'Credit_Score': ['Good', 'Good', 'Bad', 'Bad', 'Good', 'Good', 'Bad',
'Bad', 'Good', 'Bad'],
        'Loan': ['Yes', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'Yes',
'No']
    }
    df = pd.DataFrame(data)
    return df

def preprocess_data(df):
    le = LabelEncoder()
    for column in ['Job', 'Credit_Score', 'Loan']:
        df[column] = le.fit_transform(df[column])
    X = df[['Age', 'Job', 'Income', 'Credit_Score']]
    y = df['Loan']
    return train_test_split(X, y, test_size=0.3, random_state=42), le

def train_and_evaluate(X_train, X_test, y_train, y_test, label_encoder):
    model = GaussianNB()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    acc = accuracy_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)
    print(f"\nAccuracy: {acc:.2f}")
    print("\nClassification Report:\n", classification_report(y_test, y_pred,
target_names=label_encoder.classes_))
    return cm, label_encoder.classes_

def visualize_confusion_matrix(cm, class_names):
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=class_names, yticklabels=class_names)
    plt.title("Naive Bayes - Confusion Matrix for Loan Prediction")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()

def main():
    print("=== Bank Loan Prediction using Naive Bayes ===")
    df = load_data()
    (X_train, X_test, y_train, y_test), label_encoder = preprocess_data(df)
    cm, class_names = train_and_evaluate(X_train, X_test, y_train, y_test,
label_encoder)
    visualize_confusion_matrix(cm, class_names)

if __name__ == "__main__":
    main()
```