```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.metrics import accuracy_score
```

```
# Sample dataset
data = pd.DataFrame({
    'age': [25, 35, 45, 33, 50, 40, 60, 48, 55, 38],
    'income': [30000, 50000, 80000, 45000, 70000, 62000, 90000, 75000, 85000, 55000],
    'education': ['High School', 'Bachelors', 'Masters', 'PhD', 'Bachelors', 'Masters', 'PhD', 'Bachelors', 'Masters', 'High School'],
    'gender': ['Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male', 'Female'],
    'owns_house': [1, 0, 1, 0, 1, 0, 1, 0, 1, 0],  # Binary column
    'purchased_product': [0, 1, 1, 0, 1, 0, 1, 0, 1, 0]  # Target variable
})
data
```

	age	income	education	gender	owns_house	purchased_product
0	25	30000	High School	Male	1	0
1	35	50000	Bachelors	Female	0	1
2	45	80000	Masters	Male	1	1
3	33	45000	PhD	Female	0	0
4	50	70000	Bachelors	Male	1	1
5	40	62000	Masters	Female	0	0
6	60	90000	PhD	Male	1	1
7	48	75000	Bachelors	Female	0	0
8	55	85000	Masters	Male	1	1
9	38	55000	High School	Female	0	0

Next steps: Generate code with data View recommended plots New interactive sheet

```
# Splitting into features (X) and target variable (y)
X = data.drop(columns=['purchased_product'])
y = data['purchased_product']

# Encoding categorical features
label_enc = LabelEncoder()
X['education'] = label_enc.fit_transform(X['education'])  # Categorical: Convert to numbers
X['gender'] = label_enc.fit_transform(X['gender'])        # Binary: Convert to 0 and 1

# Splitting dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Gaussian Naïve Bayes (for numerical data: age, income)
scaler = StandardScaler()
X_train_gnb = scaler.fit_transform(X_train[['age', 'income']])
X_test_gnb = scaler.transform(X_test[['age', 'income']])

gnb = GaussianNB()
gnb.fit(X_train_gnb, y_train)
probs_gnb = gnb.predict_proba(X_test_gnb)  # Get probability outputs

# Multinomial Naïve Bayes (for categorical data: education)
mnb = MultinomialNB()
mnb.fit(X_train[['education']], y_train)
probs_mnb = mnb.predict_proba(X_test[['education']])

# Bernoulli Naïve Bayes (for binary data: gender, owns_house)
bnb = BernoulliNB()
bnb.fit(X_train[['gender', 'owns_house']], y_train)
probs_bnb = bnb.predict_proba(X_test[['gender', 'owns_house']])
```

```
# **Ensemble Using Probability Multiplication**
ensemble_probs = probs_gnb * probs_mnb * probs_bnb  # Multiply probabilities
ensemble_probs = ensemble_probs / ensemble_probs.sum(axis=1, keepdims=True)  # Normalize

# Final predictions
final_predictions = np.argmax(ensemble_probs, axis=1)

# Accuracy of the ensemble model
ensemble_accuracy = accuracy_score(y_test, final_predictions)

# Print accuracies
print("Gaussian Naïve Bayes Accuracy:", accuracy_score(y_test, np.argmax(probs_gnb, axis=1)))
print("Multinomial Naïve Bayes Accuracy:", accuracy_score(y_test, np.argmax(probs_mnb, axis=1)))
print("Bernoulli Naïve Bayes Accuracy:", accuracy_score(y_test, np.argmax(probs_bnb, axis=1)))
print("Ensemble Model Accuracy:", ensemble_accuracy)
```

```
Gaussian Naïve Bayes Accuracy: 0.6666666666666666
Multinomial Naïve Bayes Accuracy: 0.3333333333333333
Bernoulli Naïve Bayes Accuracy: 0.6666666666666666
Ensemble Model Accuracy: 0.6666666666666666
```

```
# Convert probabilities to DataFrame for better understanding in the form of table
prob_df = pd.DataFrame({
    'Actual Target': y_test.values,
    'GNB - P(No)': probs_gnb[:, 0], 'GNB - P(Yes)': probs_gnb[:, 1],
    'MNB - P(No)': probs_mnb[:, 0], 'MNB - P(Yes)': probs_mnb[:, 1],
    'BNB - P(No)': probs_bnb[:, 0], 'BNB - P(Yes)': probs_bnb[:, 1],
    'Ensemble - P(No)': ensemble_probs[:, 0], 'Ensemble - P(Yes)': ensemble_probs[:, 1],
    'Final Prediction': final_predictions
})

# Map 0 -> "No", 1 -> "Yes" for better readability
prob_df['Actual Target'] = prob_df['Actual Target'].map({0: "No", 1: "Yes"})
prob_df['Final Prediction'] = prob_df['Final Prediction'].map({0: "No", 1: "Yes"})

prob_df
```

	Actual Target	GNB - P(No)	GNB - P(Yes)	MNB - P(No)	MNB - P(Yes)	BNB - P(No)	BNB - P(Yes)	Ensemble - P(No)	Ensemble - P(Yes)	Final Prediction
0	Yes	0.006089	0.993911	0.571429	0.428571	0.187970	0.812030	0.001887	0.998113	Yes
1	Yes	0.999933	0.000067	0.571429	0.428571	0.936768	0.063232	0.999997	0.000003	No
2	No	0.958983	0.041017	0.571429	0.428571	0.936768	0.063232	0.997839	0.002161	No

Next steps: Generate code with prob_df View recommended plots New interactive sheet