```python
# importing all required libraries

import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, f1_score, precision_score,
recall_score

# Load the datasets

# Banknote Authentication Dataset

banknote_data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-
learning-databases/00267/data_banknote_authentication.txt',
header=None, names=['variance', 'skewness', 'curtosis', 'entropy',
'class'])
X_banknote = banknote_data.drop('class', axis=1)
y_banknote = banknote_data['class']

print("\n Banknote Dataset Description:")
print(f"  Number of features: {X_banknote.shape[1]}")
print(f"  Number of instances: {X_banknote.shape[0]}")
print(f"  Number of classes: {len(y_banknote.unique())}")
print("  Features:", list(X_banknote.columns))
```

```
 Banknote Dataset Description:
  Number of features: 4
  Number of instances: 1372
  Number of classes: 2
  Features: ['variance', 'skewness', 'curtosis', 'entropy']
```

```python
# Haberman Dataset

haberman_data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-
learning-databases/haberman/haberman.data', header=None, names=['age',
'year', 'nodes', 'survival'])
X_haberman = haberman_data.drop('survival', axis=1)
y_haberman = haberman_data['survival'].map({1: 1, 2: 0})

# Dataset Description
print("\n Haberman Dataset Description:")
print(f"  Number of features: {X_haberman.shape[1]}")
print(f"  Number of instances: {X_haberman.shape[0]}")
print(f"  Number of classes: {len(y_haberman.unique())}")
print("  Features:", list(X_haberman.columns))
```

```
 Haberman Dataset Description:
  Number of features: 3
  Number of instances: 306
  Number of classes: 2
  Features: ['age', 'year', 'nodes']
```

```
# Preprocessing datasets

scaler_banknote_data = StandardScaler()
X_banknote_scaled_data =
scaler_banknote_data.fit_transform(X_banknote)

scaler_haberman_data = StandardScaler()
X_haberman_scaled_data =
scaler_haberman_data.fit_transform(X_haberman)

# Train-test split with 80% train data and 20% test data

X_banknote_train, X_banknote_test, y_banknote_train, y_banknote_test =
train_test_split(X_banknote_scaled_data, y_banknote, test_size=0.2,
random_state=42, stratify=y_banknote)
X_haberman_train, X_haberman_test, y_haberman_train, y_haberman_test =
train_test_split(X_haberman_scaled_data, y_haberman, test_size=0.2,
random_state=42, stratify=y_haberman)

# Model Training and Evaluation

results = {}

results['banknote'] = {}

# Banknote Authentication Models

# 1. Naive Bayes

print("Banknote Authentication Dataset \n")
print("Training Naive Bayes \n")

nb_banknote = GaussianNB()
nb_banknote.fit(X_banknote_train, y_banknote_train)
y_banknote_pred_nb = nb_banknote.predict(X_banknote_test)
results['banknote']['Naive Bayes'] = {
    'Accuracy': round(accuracy_score(y_banknote_test,
y_banknote_pred_nb), 4),
    'Macro F1-Score': round(f1_score(y_banknote_test,
y_banknote_pred_nb, average='macro'), 4),
    'Macro Precision': round(precision_score(y_banknote_test,
y_banknote_pred_nb, average='macro'), 4),
    'Macro Recall': round(recall_score(y_banknote_test,
y_banknote_pred_nb, average='macro'), 4)
```

```
}

print(f"Naive Bayes Results: {results['banknote']['Naive Bayes']} \n")
print("\n---------------------------------------------------------\n")

# 2. Logistic Regression

print("Training Logistic Regression \n")

lr_banknote = LogisticRegression(random_state=42)
param_grid_lr_banknote = {'C': [0.001, 0.01, 0.1, 1, 10, 100],
'solver': ['liblinear']}
grid_search_lr_banknote = GridSearchCV(lr_banknote,
param_grid_lr_banknote, cv=5, scoring='f1_macro')
grid_search_lr_banknote.fit(X_banknote_train, y_banknote_train)
best_lr_banknote = grid_search_lr_banknote.best_estimator_
y_banknote_pred_lr = best_lr_banknote.predict(X_banknote_test)
results['banknote']['Logistic Regression'] = {
    'Accuracy': round(accuracy_score(y_banknote_test,
y_banknote_pred_lr), 4),
    'Macro F1-Score': round(f1_score(y_banknote_test,
y_banknote_pred_lr, average='macro'), 4),
    'Macro Precision': round(precision_score(y_banknote_test,
y_banknote_pred_lr, average='macro'), 4),
    'Macro Recall': round(recall_score(y_banknote_test,
y_banknote_pred_lr, average='macro'), 4)
}

print(f"Logistic Regression Results: {results['banknote']['Logistic
Regression']} \n")
print(f"Best Logistic Regression parameters:
{grid_search_lr_banknote.best_params_} \n")
print("\n---------------------------------------------------------\n")

# 3. Support Vector Machine (SVM)

print("Training SVM \n")

svm_banknote = SVC(random_state=42)
param_grid_svm_banknote = {'C': [0.1, 1, 10, 100], 'kernel':
['linear', 'rbf'], 'gamma': ['scale', 'auto']}
grid_search_svm_banknote = GridSearchCV(svm_banknote,
param_grid_svm_banknote, cv=5, scoring='f1_macro')
grid_search_svm_banknote.fit(X_banknote_train, y_banknote_train)
best_svm_banknote = grid_search_svm_banknote.best_estimator_
y_banknote_pred_svm = best_svm_banknote.predict(X_banknote_test)
results['banknote']['SVM'] = {
    'Accuracy': round(accuracy_score(y_banknote_test,
y_banknote_pred_svm), 4),
    'Macro F1-Score': round(f1_score(y_banknote_test,
y_banknote_pred_svm, average='macro'), 4),
    'Macro Precision': round(precision_score(y_banknote_test,
y_banknote_pred_svm, average='macro'), 4),
    'Macro Recall': round(recall_score(y_banknote_test,
y_banknote_pred_svm, average='macro'), 4)
}
print(f"SVM Results: {results['banknote']['SVM']} \n")
print(f"Best SVM parameters: {grid_search_svm_banknote.best_params_} \
n")
print("\n---------------------------------------------------------\n")

print("Haberman Dataset \n")

results['haberman'] = {}

# 1. Naive Bayes

print("Training Naive Bayes \n")

nb_haberman = GaussianNB()
nb_haberman.fit(X_haberman_train, y_haberman_train)
y_haberman_pred_nb = nb_haberman.predict(X_haberman_test)
results['haberman']['Naive Bayes'] = {
    'Accuracy': round(accuracy_score(y_haberman_test,
y_haberman_pred_nb), 4),
    'Macro F1-Score': round(f1_score(y_haberman_test,
y_haberman_pred_nb, average='macro'), 4),
    'Macro Precision': round(precision_score(y_haberman_test,
y_haberman_pred_nb, average='macro'), 4),
    'Macro Recall': round(recall_score(y_haberman_test,
y_haberman_pred_nb, average='macro'), 4)
}

print(f"Naive Bayes Results: {results['haberman']['Naive Bayes']} \n")
print("\n---------------------------------------------------------\n")

# 2. Logistic Regression

print("Training Logistic Regression \n")

lr_haberman = LogisticRegression(random_state=42)
param_grid_lr_haberman = {'C': [0.001, 0.01, 0.1, 1, 10, 100],
'solver': ['liblinear']}
grid_search_lr_haberman = GridSearchCV(lr_haberman,
param_grid_lr_haberman, cv=5, scoring='f1_macro')
grid_search_lr_haberman.fit(X_haberman_train, y_haberman_train)
best_lr_haberman = grid_search_lr_haberman.best_estimator_
y_haberman_pred_lr = best_lr_haberman.predict(X_haberman_test)
results['haberman']['Logistic Regression'] = {
    'Accuracy': round(accuracy_score(y_haberman_test,
y_haberman_pred_lr), 4),
    'Macro F1-Score': round(f1_score(y_haberman_test,
y_haberman_pred_lr, average='macro'), 4),
    'Macro Precision': round(precision_score(y_haberman_test,
y_haberman_pred_lr, average='macro'), 4),
    'Macro Recall': round(recall_score(y_haberman_test,
y_haberman_pred_lr, average='macro'), 4)
}

print(f"Logistic Regression Results: {results['haberman']['Logistic
Regression']} \n")
print(f"Best Logistic Regression parameters:
{grid_search_lr_haberman.best_params_} \n")
print("\n---------------------------------------------------------\n")

# 3. Support Vector Machine (SVM)

print("Training SVM \n")

svm_haberman = SVC(random_state=42)
param_grid_svm_haberman = {'C': [0.1, 1, 10, 100], 'kernel':
['linear', 'rbf'], 'gamma': ['scale', 'auto']}
grid_search_svm_haberman = GridSearchCV(svm_haberman,
param_grid_svm_haberman, cv=5, scoring='f1_macro')
grid_search_svm_haberman.fit(X_haberman_train, y_haberman_train)
best_svm_haberman = grid_search_svm_haberman.best_estimator_
y_haberman_pred_svm = best_svm_haberman.predict(X_haberman_test)
results['haberman']['SVM'] = {
    'Accuracy': round(accuracy_score(y_haberman_test,
y_haberman_pred_svm), 4),
    'Macro F1-Score': round(f1_score(y_haberman_test,
y_haberman_pred_svm, average='macro'), 4),
    'Macro Precision': round(precision_score(y_haberman_test,
y_haberman_pred_svm, average='macro'), 4),
    'Macro Recall': round(recall_score(y_haberman_test,
y_haberman_pred_svm, average='macro'), 4)
}

print(f"SVM Results: {results['haberman']['SVM']} \n")
print(f"Best SVM parameters: {grid_search_svm_haberman.best_params_} \
n")
print("\n---------------------------------------------------------\n")
```

```
Banknote Authentication Dataset

Training Naive Bayes

Naive Bayes Results: {'Accuracy': 0.8582, 'Macro F1-Score': 0.856,
'Macro Precision': 0.8571, 'Macro Recall': 0.8551}
```

```
---------------------------------------------------------

Training Logistic Regression

Logistic Regression Results: {'Accuracy': 0.9855, 'Macro F1-Score':
0.9853, 'Macro Precision': 0.9841, 'Macro Recall': 0.9869}

Best Logistic Regression parameters: {'C': 100, 'solver': 'liblinear'}



---------------------------------------------------------

Training SVM

SVM Results: {'Accuracy': 1.0, 'Macro F1-Score': 1.0, 'Macro
Precision': 1.0, 'Macro Recall': 1.0}

Best SVM parameters: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}


---------------------------------------------------------

Haberman Dataset

Training Naive Bayes

Naive Bayes Results: {'Accuracy': 0.7581, 'Macro F1-Score': 0.57,
'Macro Precision': 0.686, 'Macro Recall': 0.572}


---------------------------------------------------------

Training Logistic Regression

Logistic Regression Results: {'Accuracy': 0.7581, 'Macro F1-Score':
0.5338, 'Macro Precision': 0.7147, 'Macro Recall': 0.5516}

Best Logistic Regression parameters: {'C': 0.001, 'solver':
'liblinear'}


---------------------------------------------------------

Training SVM

SVM Results: {'Accuracy': 0.7258, 'Macro F1-Score': 0.5127, 'Macro
Precision': 0.5772, 'Macro Recall': 0.5299}

Best SVM parameters: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
```

```
---------------------------------------------------------
```