Machine Learning Programs for Dataset Handling and Classification Tasks

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_breast_cancer, load_iris
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler, LabelEncoder

# Section 3: Creation, Loading and Normalization
print("\n--- Section 3: Dataset Creation and Normalization ---")
iris = load_iris()
df_iris = pd.DataFrame(iris.data, columns=iris.feature_names)
scaler = StandardScaler()
scaled_iris = scaler.fit_transform(df_iris)

# Section 4: Breast Cancer Classification using Euclidean Distance
print("\n--- Section 4: Breast Cancer Classification ---")
cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.3,
random_state=42)
knn = KNeighborsClassifier(metric='euclidean')
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
acc = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
print("Accuracy:", acc)
print("Confusion Matrix:\n", cm)
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))

# 10-fold CV
kf = KFold(n_splits=10)
```

```
scores = cross_val_score(knn, cancer.data, cancer.target, cv=kf)
print("10-fold CV Std Deviation:", scores.std())

# Section 5: Linear Regression
print("\n--- Section 5: Linear Regression ---")
# Assuming 'linear_data.csv' exists with numeric features and target
# df = pd.read_csv('linear_data.csv')
# X, y = df.iloc[:, :-1], df.iloc[:, -1]
# linreg = LinearRegression().fit(X, y)
# y_pred = linreg.predict(X)
# print("Prediction Accuracy:", linreg.score(X, y))

# Section 6: Logistic Regression
print("\n--- Section 6: Logistic Regression ---")
# df = pd.read_csv('logistic_data.csv')
# X, y = df.iloc[:, :-1], df.iloc[:, -1]
# logreg = LogisticRegression().fit(X, y)
# print("Classification Accuracy:", logreg.score(X, y))

# Section 7: Naive Bayes Text Classification
print("\n--- Section 7: Naive Bayes ---")
# df = pd.read_csv('docs.csv')
# X = df['text']
# y = df['label']
# from sklearn.feature_extraction.text import CountVectorizer
# X_vec = CountVectorizer().fit_transform(X)
# model = MultinomialNB().fit(X_vec, y)
# y_pred = model.predict(X_vec)
# print("Accuracy:", accuracy_score(y, y_pred))
# print("Precision:", precision_score(y, y_pred, average='macro'))
# print("Recall:", recall_score(y, y_pred, average='macro'))

# Section 8: KNN (Already Done in Section 4)

# Section 9: Feature Selection (Step Forward & Backward Elimination)
print("\n--- Section 9: Feature Selection ---")
X = cancer.data
y = cancer.target
dt = DecisionTreeClassifier()
sfs = SequentialFeatureSelector(dt, direction='forward', n_features_to_select=5)
sfs.fit(X, y)
print("Selected features (forward):", sfs.get_support())

# Section 10: PCA vs LDA
```

```
print("\n--- Section 10: Dimensionality Reduction ---")
pca = PCA(n_components=2)
pca_data = pca.fit_transform(cancer.data)
lda = LinearDiscriminantAnalysis(n_components=1)
lda_data = lda.fit_transform(cancer.data, cancer.target)

# Plot PCA and LDA
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.scatter(pca_data[:, 0], pca_data[:, 1], c=cancer.target, cmap='viridis')
plt.title("PCA - 2D")
plt.subplot(1, 2, 2)
plt.scatter(lda_data, np.zeros_like(lda_data), c=cancer.target, cmap='viridis')
plt.title("LDA - 1D")
plt.tight_layout()
plt.show()

# Section 11: Decision Tree ID3
print("\n--- Section 11: Decision Tree ID3 ---")
dt = DecisionTreeClassifier(criterion='entropy')
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred))

# Section 12: Compare DT and SVM
print("\n--- Section 12: DT vs SVM ---")
svm = SVC(kernel='linear')
svm.fit(X_train, y_train)
y_svm_pred = svm.predict(X_test)
print("SVM Accuracy:", accuracy_score(y_test, y_svm_pred))

# Section 13: SVM Kernels
print("\n--- Section 13: SVM Kernel Comparison ---")
kernels = ['linear', 'rbf', 'poly']
for kernel in kernels:
    model = SVC(kernel=kernel)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"Kernel={kernel}: Accuracy = {accuracy_score(y_test, y_pred):.2f}")

# Visual Example of One SVM Kernel
plt.figure()
svm_rbf = SVC(kernel='rbf').fit(X_train[:, :2], y_train)
y_pred = svm_rbf.predict(X_test[:, :2])
```

```
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_pred, cmap='coolwarm')
plt.title("SVM RBF Kernel Classification")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

SVM Classifier with Plot

```python
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Load dataset
iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train SVM
clf = SVC(kernel='linear')
clf.fit(X_train, y_train)

# Predict & Accuracy
y_pred = clf.predict(X_test)
print("SVM Accuracy:", accuracy_score(y_test, y_pred))
```

```
# Plot
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_pred, cmap='viridis')
plt.title("SVM Classification")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.grid(True)
plt.show()
```

PCA Visualization
```python
from sklearn.decomposition import PCA

# Reduce to 2D
pca = PCA(n_components=2)
X_pca = pca.fit_transform(iris.data)

plt.scatter(X_pca[:, 0], X_pca[:, 1], c=iris.target, cmap='viridis')
plt.title("PCA - 2D Projection")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.grid(True)
plt.show()
```

LDA Visualization
```python
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

lda = LDA(n_components=2)
X_lda = lda.fit_transform(iris.data, iris.target)

plt.scatter(X_lda[:, 0], X_lda[:, 1], c=iris.target, cmap='viridis')
plt.title("LDA - 2D Projection")
plt.xlabel("LD1")
plt.ylabel("LD2")
plt.grid(True)
plt.show()
```

Decision Tree Classifier with Plot
```python
from sklearn.tree import DecisionTreeClassifier, plot_tree
```

```
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred))

# Plot tree
plt.figure(figsize=(10, 6))
plot_tree(clf, filled=True, feature_names=iris.feature_names[:2],
class_names=iris.target_names)
plt.title("Decision Tree")
plt.show()
```

Ridge Regression with Plot
```python
from sklearn.linear_model import Ridge
import numpy as np

# Synthetic Data
np.random.seed(42)
X = np.random.rand(100, 1) * 10
y = 3 * X.squeeze() + np.random.randn(100) * 2

ridge = Ridge(alpha=1.0)
ridge.fit(X, y)
y_pred = ridge.predict(X)

plt.scatter(X, y, color='blue')
plt.plot(X, y_pred, color='red')
plt.title("Ridge Regression")
plt.xlabel("X")
plt.ylabel("y")
plt.grid(True)
plt.show()
```

Lasso Regression with Plot
```python
from sklearn.linear_model import Lasso

lasso = Lasso(alpha=0.1)
lasso.fit(X, y)
y_pred = lasso.predict(X)
```

```
plt.scatter(X, y, color='green')
plt.plot(X, y_pred, color='black')
plt.title("Lasso Regression")
plt.xlabel("X")
plt.ylabel("y")
plt.grid(True)
plt.show()
```