ASSIGNMENT 8

Title: Apply different feature selection approaches for the classification/regression task. Compare the performance of different feature selection approach.

Objective: The objective of this lab assignment is to explore various feature selection techniques for classification and regression tasks

Dataset: Use the UCI Iris dataset for the classification task and the Boston Housing dataset for the regression task.

Tasks:
1) Load the Iris dataset and the Boston Housing dataset.
2) Preprocess the data: handle missing values, encode categorical variables (if any), and normalize/standardize features.
3) Split each dataset into features (X) and target variable (y).
4) For each dataset and each feature selection approach (minimum 3), follow these steps: a. Apply the feature selection technique to select a subset of features. b. Split the data into training and testing sets (e.g., 70% training, 30% testing). c. Train a classification model (e.g., Logistic Regression, Random Forest) for the Iris dataset and a regression model (e.g., Linear Regression, Decision Tree) for the Boston Housing dataset using the selected features. d. Evaluate the model's performance on the testing set using appropriate metrics (e.g., accuracy, mean squared error).
5) Compare and analyze the performance of each approach in terms of model performance and the number of selected features. 6) Summarize your findings and insights in a report, including a comparison table or visualization.

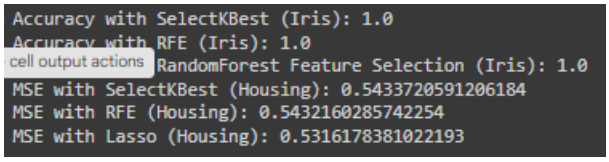
Code:

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris, fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression, LinearRegression, LassoCV
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.feature_selection import SelectKBest, mutual_info_classif, mutual_info_regression
from sklearn.feature_selection import RFE
from sklearn.metrics import accuracy_score, mean_squared_error
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor

# ------------------------------------------------------------
# Load Iris dataset
iris = load_iris()
X_iris, y_iris = iris.data, iris.target
```

```
# Standardize features
scaler = StandardScaler()
X_iris = scaler.fit_transform(X_iris)

# Load Boston Housing dataset
housing = fetch_california_housing()
X_housing, y_housing = housing.data, housing.target

# Impute any missing values (if present) and standardize features
imputer = SimpleImputer(strategy='mean')
X_housing = imputer.fit_transform(X_housing)
X_housing = scaler.fit_transform(X_housing)

# ------------------------------------------------------------
# Split data into training and testing sets
X_train_iris, X_test_iris, y_train_iris, y_test_iris = train_test_split(X_iris, y_iris, test_size=0.3, random_state=42)

# Initialize classifiers
log_reg = LogisticRegression()
rf_clf = RandomForestClassifier()

# 1. Filter Method - SelectKBest
k_best = SelectKBest(score_func=mutual_info_classif, k=2)
X_train_iris_kbest = k_best.fit_transform(X_train_iris, y_train_iris)
X_test_iris_kbest = k_best.transform(X_test_iris)

# Train and evaluate using Logistic Regression
log_reg.fit(X_train_iris_kbest, y_train_iris)
y_pred_iris_kbest = log_reg.predict(X_test_iris_kbest)
print(f"Accuracy with SelectKBest (Iris): {accuracy_score(y_test_iris, y_pred_iris_kbest)}")

# 2. Wrapper Method - Recursive Feature Elimination (RFE)
rfe = RFE(log_reg, n_features_to_select=2)
X_train_iris_rfe = rfe.fit_transform(X_train_iris, y_train_iris)
X_test_iris_rfe = rfe.transform(X_test_iris)

# Train and evaluate using Logistic Regression
log_reg.fit(X_train_iris_rfe, y_train_iris)
y_pred_iris_rfe = log_reg.predict(X_test_iris_rfe)
print(f"Accuracy with RFE (Iris): {accuracy_score(y_test_iris, y_pred_iris_rfe)}")

# 3. Embedded Method - RandomForest Feature Importance
rf_clf.fit(X_train_iris, y_train_iris)
importances = rf_clf.feature_importances_
sorted_indices = np.argsort(importances)[-2:]

X_train_iris_rf = X_train_iris[:, sorted_indices]
X_test_iris_rf = X_test_iris[:, sorted_indices]

# Train and evaluate using Logistic Regression
log_reg.fit(X_train_iris_rf, y_train_iris)
y_pred_iris_rf = log_reg.predict(X_test_iris_rf)
print(f"Accuracy with RandomForest Feature Selection (Iris): {accuracy_score(y_test_iris, y_pred_iris_rf)}")
```

```
# Split data into training and testing sets
X_train_housing, X_test_housing, y_train_housing, y_test_housing = train_test_split(X_housing, y_housing, test_size=0.3, random_state=42)

# Initialize regression models
lin_reg = LinearRegression()
rf_reg = RandomForestRegressor()

# 1. Filter Method - SelectKBest
k_best_reg = SelectKBest(score_func=mutual_info_regression, k=5)
X_train_housing_kbest = k_best_reg.fit_transform(X_train_housing, y_train_housing)
X_test_housing_kbest = k_best_reg.transform(X_test_housing)

# Train and evaluate using Linear Regression
lin_reg.fit(X_train_housing_kbest, y_train_housing)
y_pred_housing_kbest = lin_reg.predict(X_test_housing_kbest)
print(f"MSE with SelectKBest (Housing): {mean_squared_error(y_test_housing, y_pred_housing_kbest)}")

# 2. Wrapper Method - Recursive Feature Elimination (RFE)
rfe_reg = RFE(lin_reg, n_features_to_select=5)
X_train_housing_rfe = rfe_reg.fit_transform(X_train_housing, y_train_housing)
X_test_housing_rfe = rfe_reg.transform(X_test_housing)

# Train and evaluate using Linear Regression
lin_reg.fit(X_train_housing_rfe, y_train_housing)
y_pred_housing_rfe = lin_reg.predict(X_test_housing_rfe)
print(f"MSE with RFE (Housing): {mean_squared_error(y_test_housing, y_pred_housing_rfe)}")

# 3. Embedded Method - Lasso Regression
lasso = LassoCV()
lasso.fit(X_train_housing, y_train_housing)
X_train_housing_lasso = X_train_housing[:, np.abs(lasso.coef_) > 0.1]
X_test_housing_lasso = X_test_housing[:, np.abs(lasso.coef_) > 0.1]

# Train and evaluate using Linear Regression
lin_reg.fit(X_train_housing_lasso, y_train_housing)
y_pred_housing_lasso = lin_reg.predict(X_test_housing_lasso)
print(f"MSE with Lasso (Housing): {mean_squared_error(y_test_housing, y_pred_housing_lasso)}")
```

Output:


Accuracy with SelectKBest (Iris): 1.0
Accuracy with RFE (Iris): 1.0
cell output actions
RandomForest Feature Selection (Iris): 1.0
MSE with SelectKBest (Housing): 0.5433720591206184
MSE with RFE (Housing): 0.5432160285742254
MSE with Lasso (Housing): 0.5316178381022193