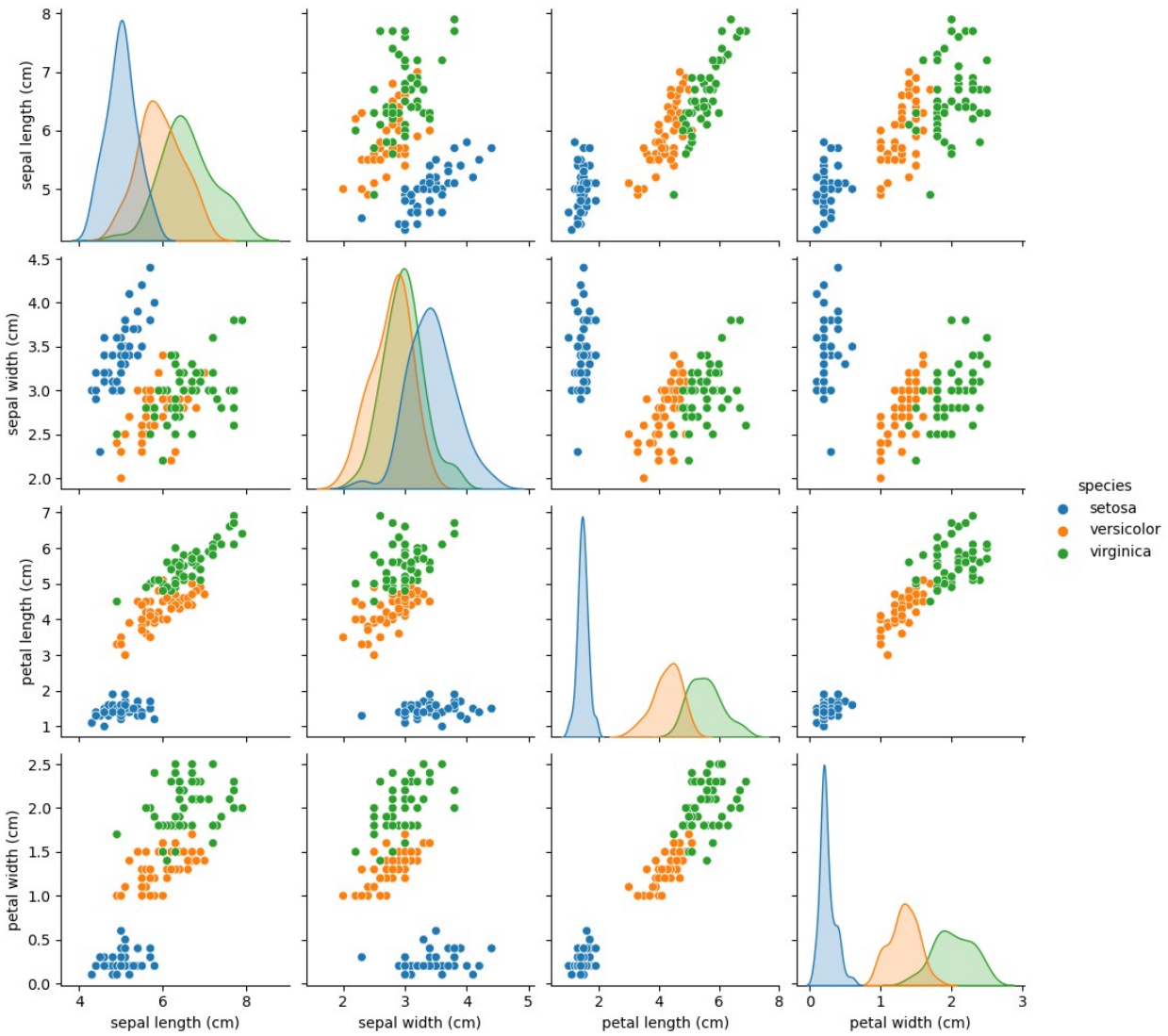
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import StandardScaler

import warnings
warnings.filterwarnings('ignore')

iris = load_iris()
X = iris.data
y = iris.target

iris_df = pd.DataFrame(X, columns=iris.feature_names)
iris_df['species'] = pd.Categorical.from_codes(y, iris.target_names)

print("Dataset Info:")
print(iris_df.info())
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype   
---  ------             --------------  -----   
 0   sepal length (cm)  150 non-null    float64 
 1   sepal width (cm)   150 non-null    float64 
 2   petal length (cm)  150 non-null    float64 
 3   petal width (cm)   150 non-null    float64 
 4   species            150 non-null    category
dtypes: category(1), float64(4)
memory usage: 5.1 KB
None
```

```
sns.pairplot(iris_df, hue='species')
plt.show()
```

```
c:\Users\Hariesh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hariesh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1057:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
```

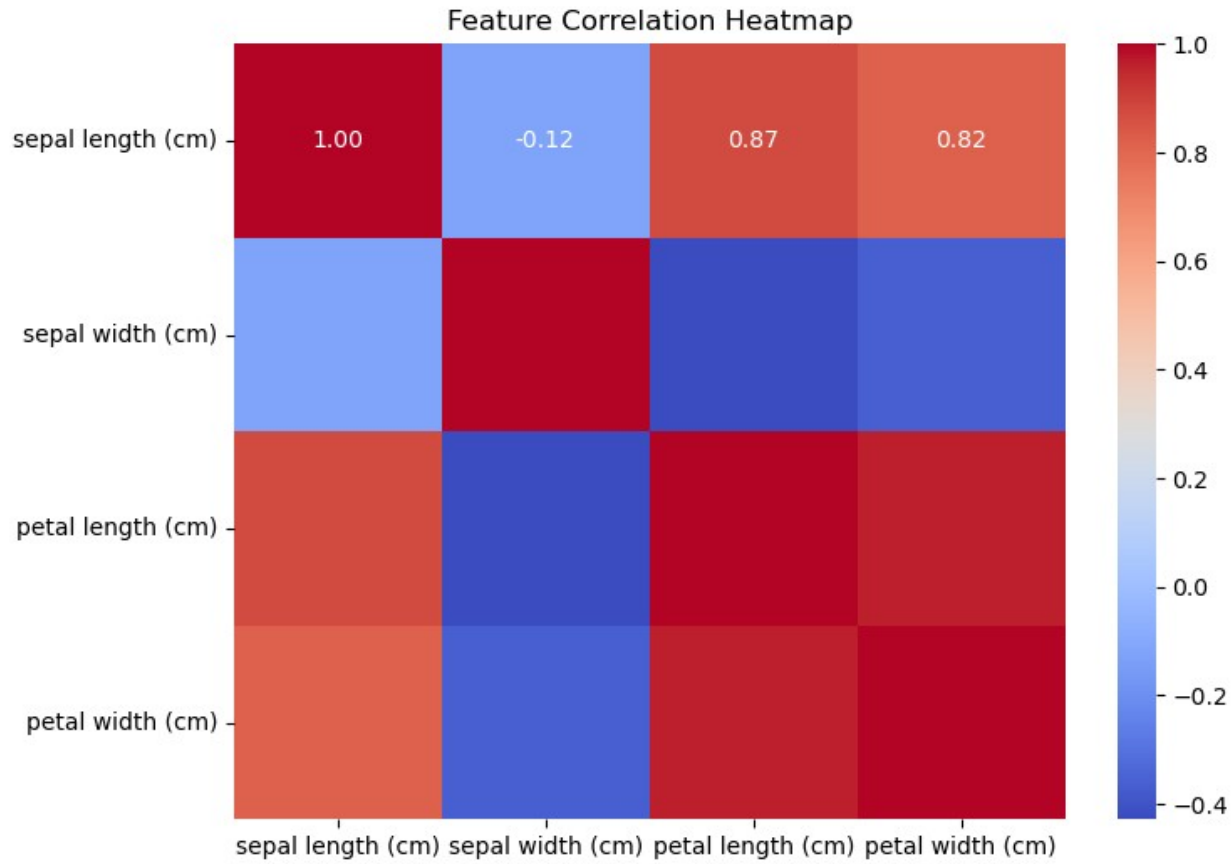
```
retain current behavior or observed=True to adopt the future default
and silence this warning.
  grouped_data = data.groupby(
c:\Users\Hariesh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hariesh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1057:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  grouped_data = data.groupby(
c:\Users\Hariesh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hariesh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1057:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  grouped_data = data.groupby(
c:\Users\Hariesh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hariesh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1057:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  grouped_data = data.groupby(
```




```
print("\nMissing values in the dataset:")
print(iris_df.isnull().sum())
```

```
Missing values in the dataset:
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
species              0
dtype: int64
```

```
plt.figure(figsize=(8,6))
sns.heatmap(iris_df.drop(columns='species').corr(), annot=True,
cmap='coolwarm', fmt='.2f')
plt.title('Feature Correlation Heatmap')
plt.show()
```




```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

dt_classifier = DecisionTreeClassifier(random_state=42)
rf_classifier = RandomForestClassifier(random_state=42)

dt_param_grid = {
    'max_depth': [3, 5, 7, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'criterion': ['gini', 'entropy']
}

dt_grid_search = GridSearchCV(dt_classifier, dt_param_grid, cv=5,
n_jobs=-1)
dt_grid_search.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(random_state=42),
n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 5, 7, None],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10]})
```

```
rf_param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 5, 7, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

rf_grid_search = GridSearchCV(rf_classifier, rf_param_grid, cv=5,
n_jobs=-1)
rf_grid_search.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
n_jobs=-1,
             param_grid={'bootstrap': [True, False],
                         'max_depth': [3, 5, 7, None],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'n_estimators': [50, 100, 200]})
```

```
print("Best Parameters for Decision Tree:")
print(dt_grid_search.best_params_)
```

```
Best Parameters for Decision Tree:
{'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 1,
'min_samples_split': 10}
```

```
print("Best Parameters for Random Forest:")
print(rf_grid_search.best_params_)
```

```
Best Parameters for Random Forest:
{'bootstrap': True, 'max_depth': 3, 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 200}
```

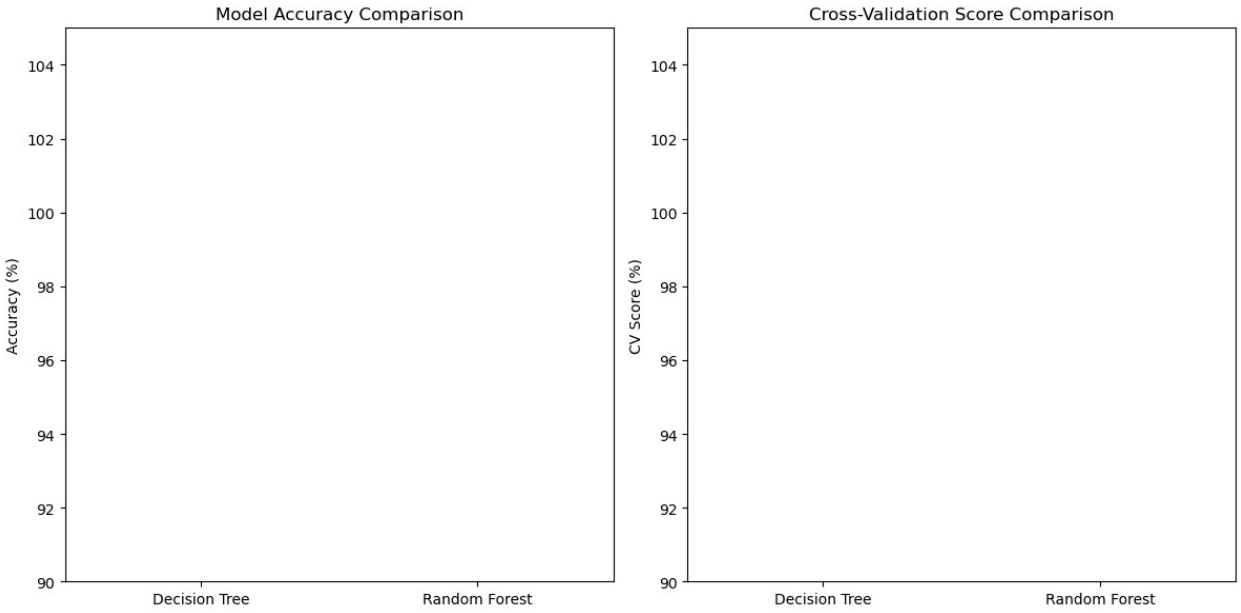
```
dt_best = dt_grid_search.best_estimator_
rf_best = rf_grid_search.best_estimator_

y_pred_dt = dt_best.predict(X_test)
y_pred_rf = rf_best.predict(X_test)

accuracy_dt = accuracy_score(y_test, y_pred_dt)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
conf_matrix_dt = confusion_matrix(y_test, y_pred_dt)
conf_matrix_rf = confusion_matrix(y_test, y_pred_rf)

cv_dt = cross_val_score(dt_best, X, y, cv=5).mean()
cv_rf = cross_val_score(rf_best, X, y, cv=5).mean()

print("\nDecision Tree Performance:")
print(f"Accuracy: {accuracy_dt * 100:.2f}%")
print(f"Confusion Matrix:\n{conf_matrix_dt}")
print(f"Cross-Validation Score: {cv_dt * 100:.2f}%\n")
```

```
Decision Tree Performance:
Accuracy: 100.00%
Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
Cross-Validation Score: 96.67%
```

```
print("Random Forest Performance:")
print(f"Accuracy: {accuracy_rf * 100:.2f}%")
print(f"Confusion Matrix:\n{conf_matrix_rf}")
print(f"Cross-Validation Score: {cv_rf * 100:.2f}%\n")
```

```
Random Forest Performance:
Accuracy: 100.00%
Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
Cross-Validation Score: 96.00%
```

```
models = ['Decision Tree', 'Random Forest']
accuracies = [accuracy_dt, accuracy_rf]
cv_scores = [cv_dt, cv_rf]

fig, ax = plt.subplots(1, 2, figsize=(12, 6))

ax[0].bar(models, accuracies, color=['blue', 'green'])
ax[0].set_title('Model Accuracy Comparison')
ax[0].set_ylabel('Accuracy (%)')
ax[0].set_ylim([90, 105])

ax[1].bar(models, cv_scores, color=['blue', 'green'])
ax[1].set_title('Cross-Validation Score Comparison')
ax[1].set_ylabel('CV Score (%)')
ax[1].set_ylim([90, 105])

plt.tight_layout()
plt.show()
```


Model Accuracy Comparison
Accuracy (%)
104
102
100
98
96
94
92
90
Decision Tree
Random Forest
Cross-Validation Score Comparison
CV Score (%)
104
102
100
98
96
94
92
90
Decision Tree
Random Forest