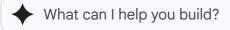
```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSe
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_se
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from scipy.stats import uniform
import warnings
warnings.filterwarnings('ignore')
```

```
# Load the Wine dataset (multi-class classification)
wine = datasets.load_wine()
X = wine.data
y = wine.target

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```



```
models = {
    "Logistic Regression": LogisticRegression(),
    "Random Forest": RandomForestClassifier(),
    "Decision Tree": DecisionTreeClassifier(),
    "SVM": SVC()
}
results = []
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    results.append({
        "Model": name,
        "Accuracy": accuracy_score(y_test, y_pred),
        "Precision": precision_score(y_test, y_pred, average='weighted'),
        "Recall": recall_score(y_test, y_pred, average='weighted'),
        "F1 Score": f1_score(y_test, y_pred, average='weighted')
    })
# Convert to DataFrame
results df = pd.DataFrame(results).sort values(by="F1 Score", ascending=False)
print(" Model Comparison:\n", results_df)
```

→ Model Comparison:

	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	1.000000	1.000000	1.000000	1.000000
1	Random Forest	1.000000	1.000000	1.000000	1.000000
3	SVM	1.000000	1.000000	1.000000	1.000000
2	Decision Tree	0.944444	0.946296	0.944444	0.943997

```
param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': [0.1, 1, 'scale', 'auto']
}

grid_svm = GridSearchCV(SVC(), param_grid, cv=5, scoring='f1_weighted')
grid_svm.fit(X_train, y_train)

print("\n GridSearchCV Best Params:", grid_svm.best_params_)
print("F1 Score with GridSearchCV:", f1_score(y_test, grid_svm.predict(X_test),
```

 $\overline{2}$

GridSearchCV Best Params: {'C': 0.1, 'gamma': 0.1, 'kernel': 'linear'} F1 Score with GridSearchCV: 0.9725248123940935



@ RandomizedSearchCV Best Params: {'n_estimators': 200, 'min_samples_split'
F1 Score with RandomizedSearchCV: 1.0

```
# Compare best tuned models
best_models = {
    "Tuned SVM": grid_svm.best_estimator_,
    "Tuned Random Forest": random_rf.best_estimator_
}

for name, model in best_models.items():
    y_pred = model.predict(X_test)
    print(f"\n \ Evaluation for {name}")
    print(classification_report(y_test, y_pred))
```

_		_
-	•	_
	→	\mathbf{v}
-	_	_

<pre>Evaluation</pre>	for Tuned	SVM		
	precision	recall	f1-score	support
0	1 00	1 00	1 00	1.4
0	1.00	1.00	1.00	14
1	1.00	0.93	0.96	14
2	0.89	1.00	0.94	8
accuracy			0.97	36
macro avg	0.96	0.98	0.97	36
weighted avg	0.98	0.97	0.97	36

<pre>Evaluation</pre>	for Tuned	Random For		
_	precision	recall	f1-score	support
0	1.00	1.00	1.00	14
1	1.00	1.00	1.00	14
2	1.00	1.00	1.00	8
accuracy			1.00	36
macro avg	1.00	1.00	1.00	36
weighted avg	1.00	1.00	1.00	36

Start coding or generate with AI.