

```
In [33]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import KFold, StratifiedKFold, cross_val_score, GridSe
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder

from sklearn.svm import SVC

from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
In [34]: data = pd.read_csv("data.csv")
data.head()
```

```
Out[34]:
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothn
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

5 rows × 33 columns

```
In [35]: data.columns
```

```
Out[35]: Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
'fractal_dimension_se', 'radius_worst', 'texture_worst',
'perimeter_worst', 'area_worst', 'smoothness_worst',
'compactness_worst', 'concavity_worst', 'concave points_worst',
'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
dtype='object')
```

```
In [36]: data = data.drop(columns=["id", "Unnamed: 32"])

le = LabelEncoder()
data["diagnosis"] = le.fit_transform(data["diagnosis"])
```

```
X = data.drop("diagnosis", axis=1)
y = data["diagnosis"]
```

```
In [37]: print("\nK-FOLD CROSS VALIDATION (Random Forest)")

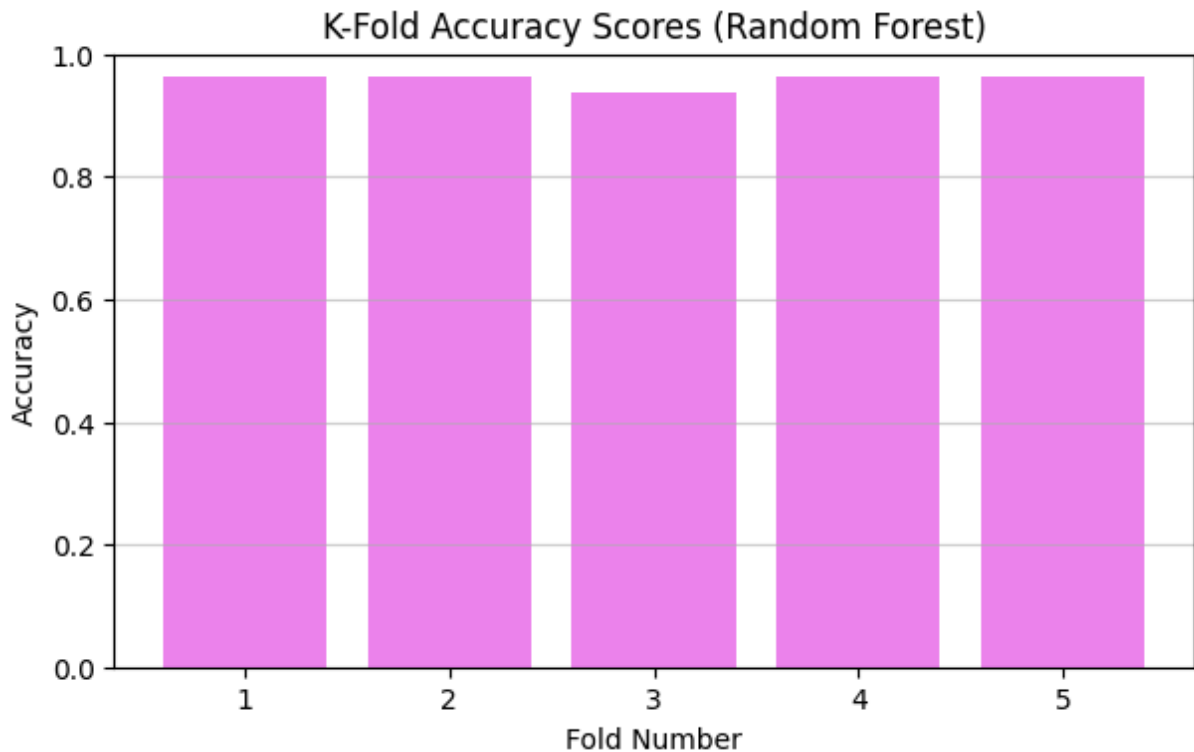
kf = KFold(n_splits=5, shuffle=True, random_state=42)
rf = RandomForestClassifier(random_state=42)

kf_scores = cross_val_score(rf, X, y, cv=kf, scoring="accuracy")

print("Accuracy Scores:", kf_scores)
print("Mean Accuracy:", kf_scores.mean())
```

K-FOLD CROSS VALIDATION (Random Forest)
Accuracy Scores: [0.96491228 0.96491228 0.93859649 0.96491228 0.96460177]
Mean Accuracy: 0.9595870206489675

```
In [62]: plt.figure(figsize=(7,4))
plt.bar(range(1, 6), kf_scores, color="violet")
plt.xlabel("Fold Number")
plt.ylabel("Accuracy")
plt.title("K-Fold Accuracy Scores (Random Forest)")
plt.ylim(0, 1)
plt.grid(axis='y', alpha=0.6)
plt.show()
```



```
In [39]: print("\nSTRATIFIED K-FOLD CROSS VALIDATION (Random Forest)")

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

skf_scores = cross_val_score(rf, X, y, cv=skf, scoring="accuracy")
```

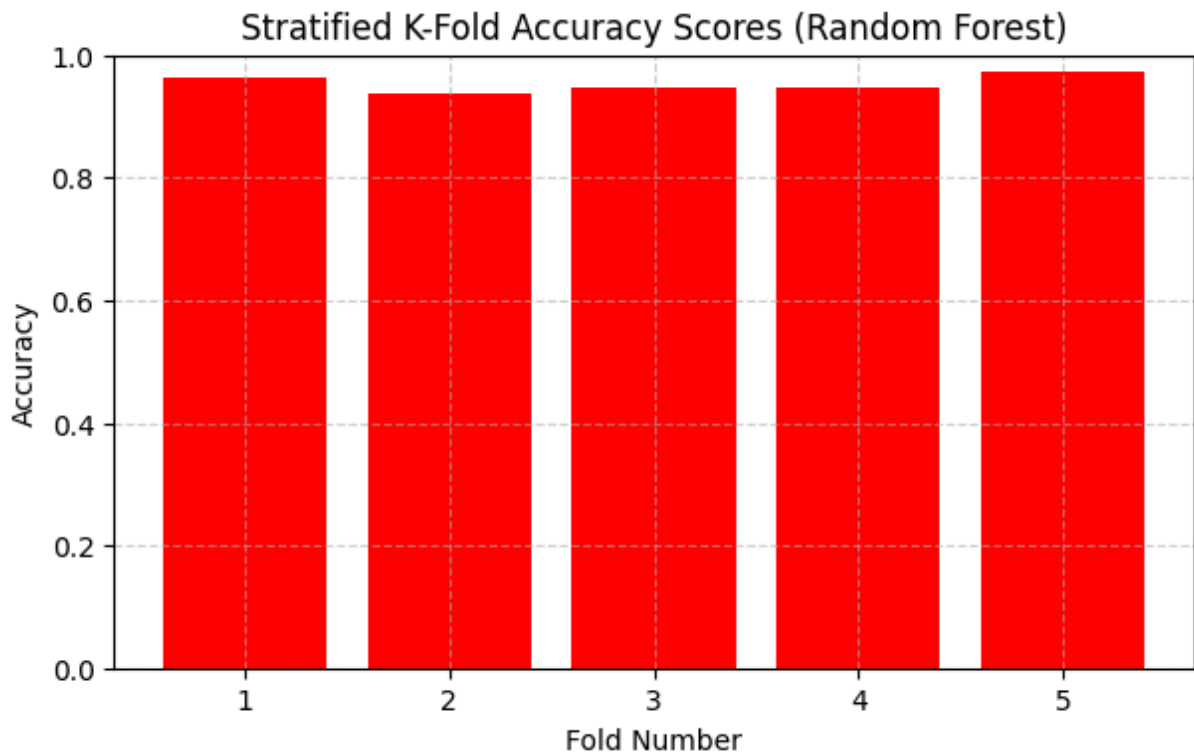
```
print("Accuracy Scores:", skf_scores)
print("Mean Accuracy:", skf_scores.mean())
```

STRATIFIED K-FOLD CROSS VALIDATION (Random Forest)

Accuracy Scores: [0.96491228 0.93859649 0.94736842 0.94736842 0.97345133]

Mean Accuracy: 0.9543393882937432

```
In [61]: plt.figure(figsize=(7, 4))
plt.bar(range(1, 6), skf_scores, color="red")
plt.xlabel("Fold Number")
plt.ylabel("Accuracy")
plt.title("Stratified K-Fold Accuracy Scores (Random Forest)")
plt.ylim(0, 1)
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```



```
In [50]: print("\nHYPERPARAMETER TUNING (Random Forest)")

param_grid = {
    "n_estimators": [100, 200, 300],
    "max_depth": [None, 10, 20],
    "min_samples_split": [2, 5, 10]
}

grid = GridSearchCV(
    RandomForestClassifier(random_state=42),
    param_grid,
    cv=5,
    scoring="accuracy"
)

grid.fit(X, y)
```

```
best_rf = grid.best_estimator_

print("Best Parameters:", grid.best_params_)
print("Best Cross-Validation Accuracy:", grid.best_score_)
```

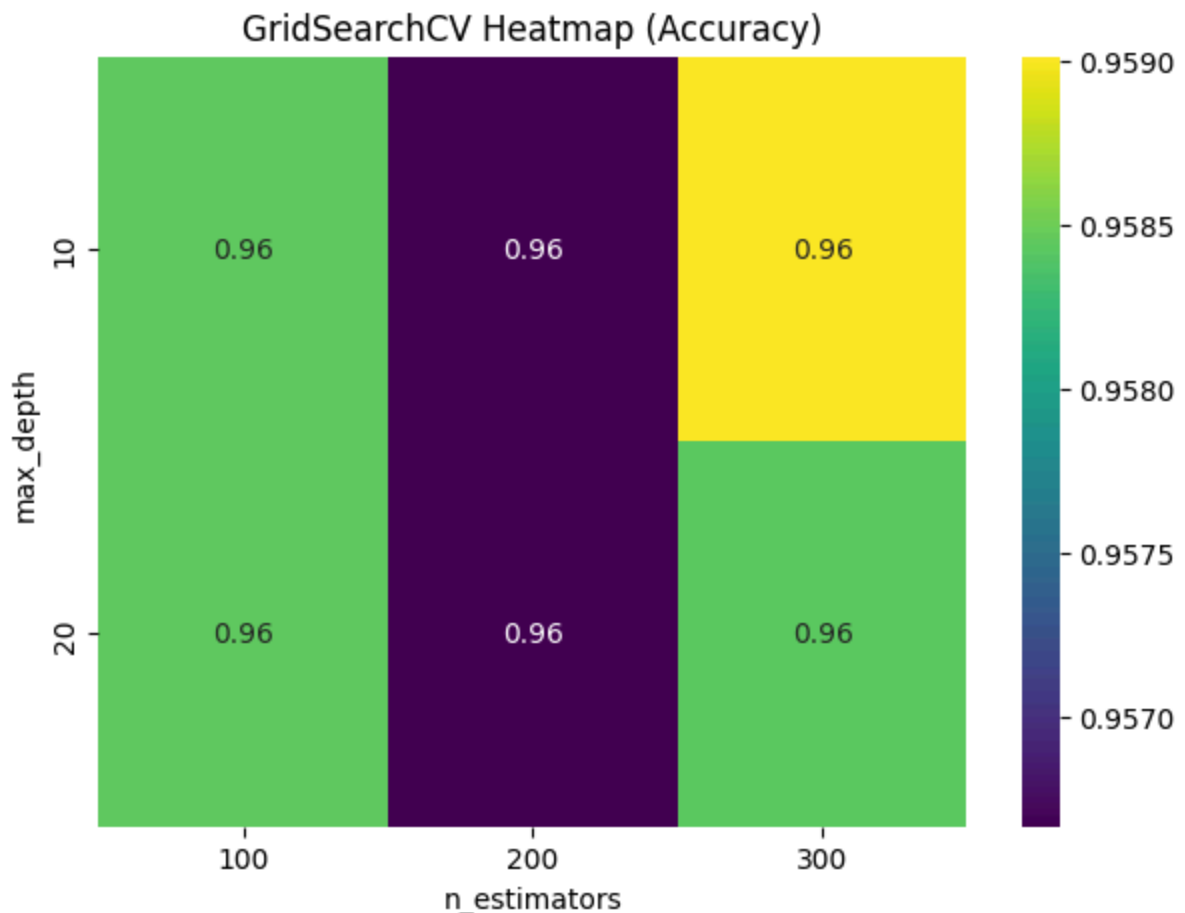
HYPERPARAMETER TUNING (Random Forest)

Best Parameters: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 300}

Best Cross-Validation Accuracy: 0.9613569321533924

```
In [51]: results = pd.DataFrame(grid.cv_results_)
pivot = results.pivot_table(values="mean_test_score",
                             index="param_max_depth",
                             columns="param_n_estimators")

plt.figure(figsize=(7,5))
sns.heatmap(pivot, annot=True, cmap='viridis')
plt.title("GridSearchCV Heatmap (Accuracy)")
plt.ylabel("max_depth")
plt.xlabel("n_estimators")
plt.show()
```



```
In [52]: print("\nMODEL COMPARISON")

models = {
    "Random Forest": best_rf,
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": Pipeline([
        ("scaler", StandardScaler()),
```

```

        ("svm", SVC(kernel="linear"))
    ])
}

for name, model in models.items():
    scores = cross_val_score(model, X, y, cv=5, scoring="accuracy")
    print(f"{name} Mean Accuracy: {scores.mean()}")

print("\nFINAL EVALUATION METRICS (Random Forest)")

best_rf.fit(X, y)
y_pred = best_rf.predict(X)

print("Accuracy :", accuracy_score(y, y_pred))
print("Precision:", precision_score(y, y_pred))
print("Recall   :", recall_score(y, y_pred))
print("F1 Score :", f1_score(y, y_pred))

```

MODEL COMPARISON

Random Forest Mean Accuracy: 0.9613569321533924
 Decision Tree Mean Accuracy: 0.9173420276354604
 SVM Mean Accuracy: 0.9718987734823784

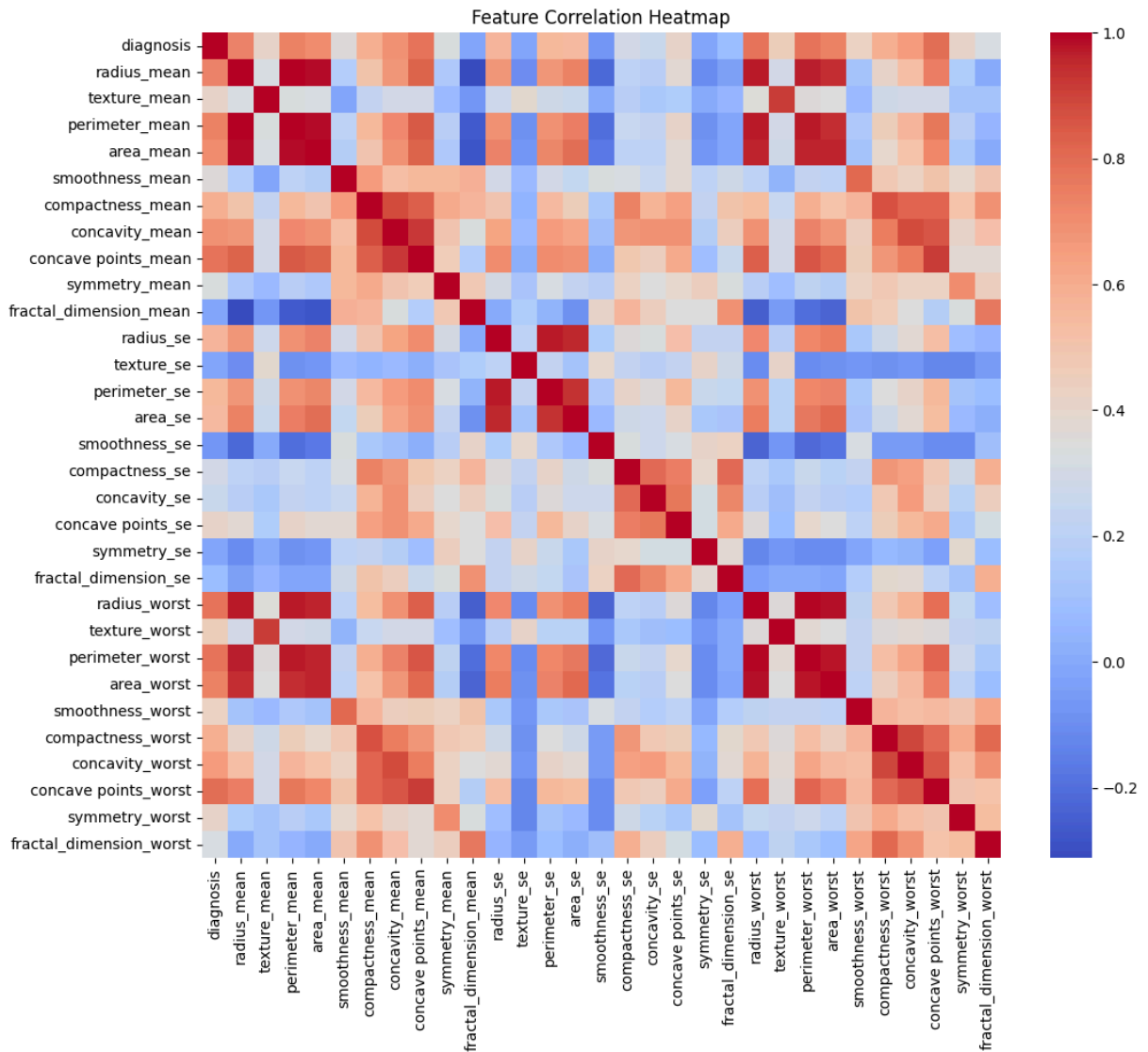
FINAL EVALUATION METRICS (Random Forest)

Accuracy : 1.0
 Precision: 1.0
 Recall : 1.0
 F1 Score : 1.0

```

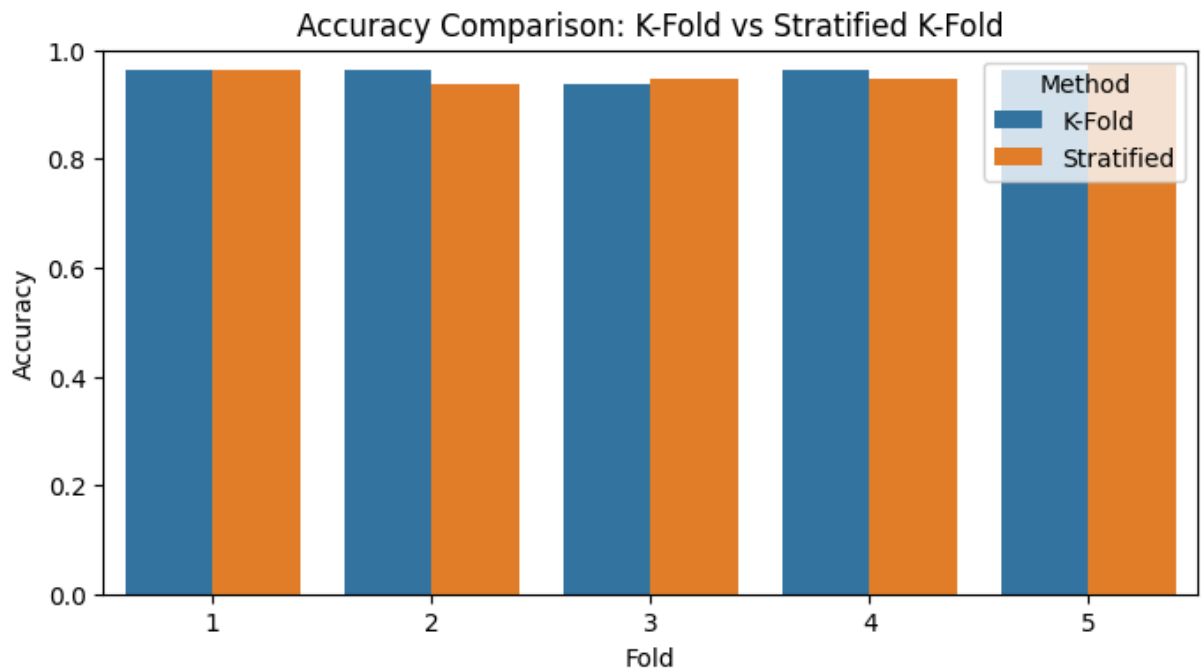
In [53]: plt.figure(figsize=(12, 10))
sns.heatmap(data.corr(), cmap='coolwarm', annot=False)
plt.title("Feature Correlation Heatmap")
plt.show()

```

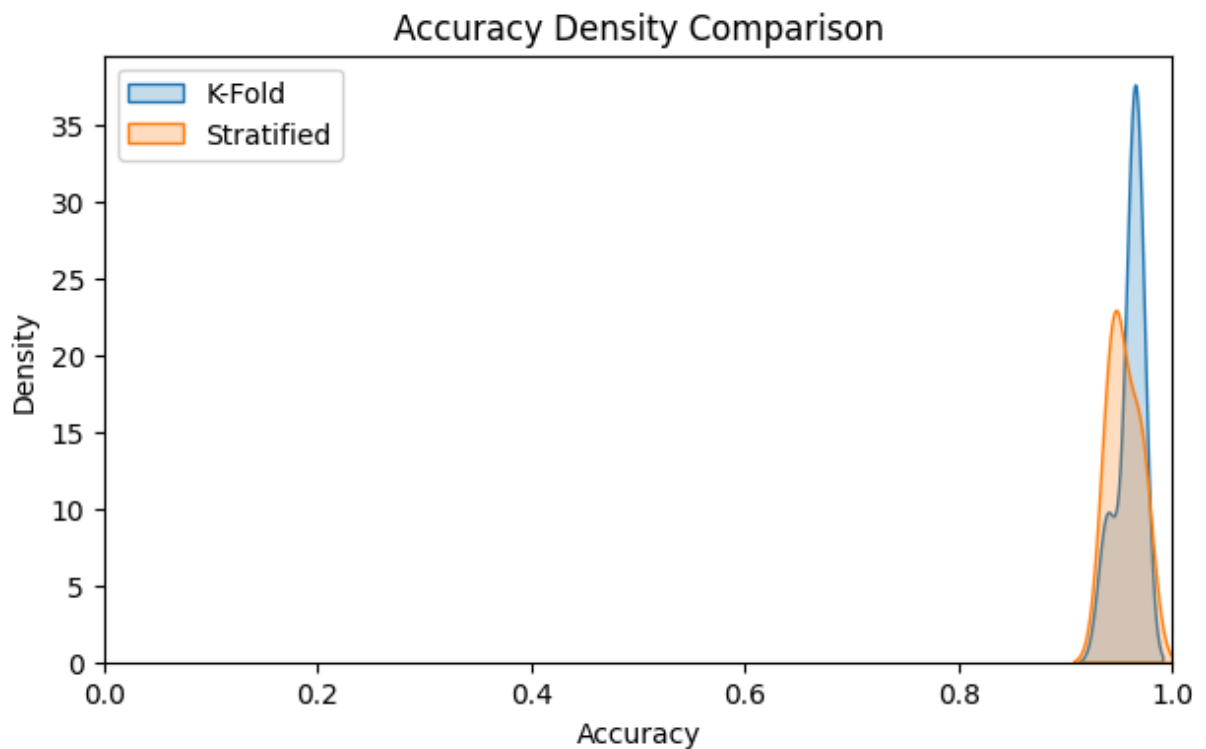


```
In [54]: df = pd.DataFrame({
    "Fold": list(range(1, 6)) * 2,
    "Accuracy": list(kf_scores) + list(skf_scores),
    "Method": ["K-Fold"] * 5 + ["Stratified"] * 5
})

plt.figure(figsize=(8, 4))
sns.barplot(data=df, x="Fold", y="Accuracy", hue="Method")
plt.ylim(0, 1)
plt.title("Accuracy Comparison: K-Fold vs Stratified K-Fold")
plt.show()
```



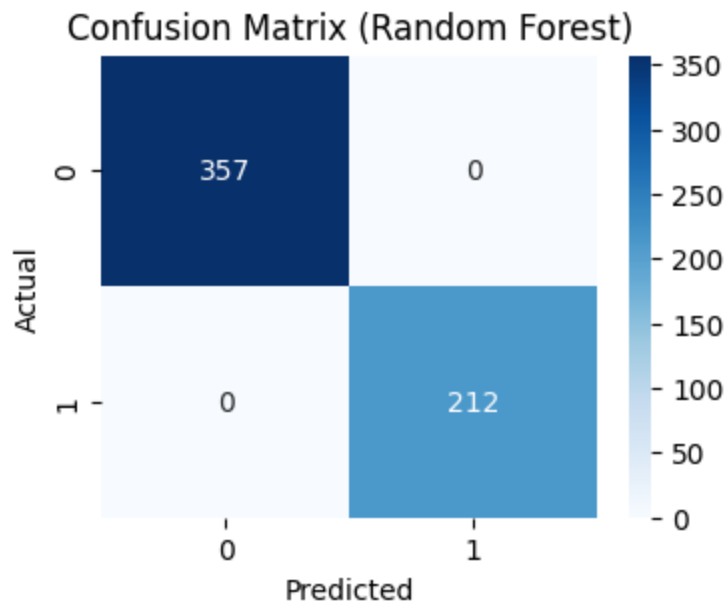
```
In [58]: plt.figure(figsize=(7, 4))
sns.kdeplot(kf_scores, fill=True, label="K-Fold")
sns.kdeplot(skf_scores, fill=True, label="Stratified")
plt.xlim(0, 1)
plt.title("Accuracy Density Comparison")
plt.xlabel("Accuracy")
plt.legend()
plt.show()
```



```
In [60]: from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(y, y_pred)

plt.figure(figsize=(4,3))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix (Random Forest)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



In []: