

[1]
✓ 1s

```
# Import libraries
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore') # Hide warnings for clean output

# Step 1: Download a dataset from Kaggle (Titanic dataset)
url = 'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv'
df = pd.read_csv(url)

# Step 2: Handle missing values appropriately
df['Age'] = df['Age'].fillna(df['Age'].median())
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])
df.drop(['Cabin', 'PassengerId', 'Name', 'Ticket'], axis=1, inplace=True)

# Encoding categorical data
df = pd.get_dummies(df, columns=['Sex', 'Embarked'], drop_first=True)

print("--- Missing values after handling ---")
print(df.isnull().sum())
```

✓

```
--- Missing values after handling ---
Survived      0
Pclass        0
Age            0
SibSp         0
Parch         0
Fare          0
Sex_male      0
Embarked_Q    0
Embarked_S    0
dtype: int64
```

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Splitting data
X = df.drop('Survived', axis=1)
y = df['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 3: Apply machine learning models with a grid search optimizer
print("--- Grid Search Evaluation ---")

# Random Forest Model
rf_params = {'n_estimators': [50, 100], 'max_depth': [None, 5, 10]}
rf_model = RandomForestClassifier(random_state=42)
rf_grid = GridSearchCV(rf_model, rf_params, cv=3, scoring='accuracy')
rf_grid.fit(X_train, y_train)
best_rf = rf_grid.best_estimator_
rf_pred = best_rf.predict(X_test)
print(f"Random Forest Accuracy: {accuracy_score(y_test, rf_pred):.3f}")

# Logistic Regression Model
lr_params = {'C': [0.1, 1, 10], 'max_iter': [200, 500]}
lr_model = LogisticRegression(random_state=42)
lr_grid = GridSearchCV(lr_model, lr_params, cv=3, scoring='accuracy')
lr_grid.fit(X_train, y_train)
best_lr = lr_grid.best_estimator_
lr_pred = best_lr.predict(X_test)
print(f"Logistic Regression Accuracy: {accuracy_score(y_test, lr_pred):.3f}")
```

✓

```
--- Grid Search Evaluation ---
Random Forest Accuracy: 0.816
Logistic Regression Accuracy: 0.804
```

```
from sklearn.feature_selection import SelectKBest, f_classif

# Step 4: Use two feature selection methods
print("--- Feature Selection ---")

# Method 1: Feature Importance
importances = best_rf.feature_importances_
feature_imp_df = pd.DataFrame({'Feature': X.columns, 'Importance': importances}).sort_values(
    'Importance', ascending=False)
print("1. Top features (Feature Importance):")
print(feature_imp_df.head(4).to_string(index=False))

# Method 2: SelectKBest
selector = SelectKBest(score_func=f_classif, k=4)
selector.fit(X_train, y_train)
selected_cols = X.columns[selector.get_support()]
print("\n2. Selected features (SelectKBest):")
print(list(selected_cols))

# Step 5: Apply machine learning models to the selected features and evaluate
X_train_selected = selector.transform(X_train)
X_test_selected = selector.transform(X_test)

best_rf.fit(X_train_selected, y_train)
final_pred = best_rf.predict(X_test_selected)

print("\n--- Final Evaluation ---")
print(f"Model Accuracy with selected features: {accuracy_score(y_test, final_pred):.3f}")
```

✓

```
--- Feature Selection ---
1. Top features (Feature Importance):
Feature  Importance
Sex_male    0.433145
Pclass      0.166810
Fare        0.154899
Age         0.114450

2. Selected features (SelectKBest):
['Pclass', 'Fare', 'Sex_male', 'Embarked_S']

--- Final Evaluation ---
Model Accuracy with selected features: 0.799
```

Start coding or [generate](#) with AI.