

```

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.feature_selection import SelectKBest, mutual_info_classif, RFE
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

DOWNLOAD DATASET .1 #
Using the Titanic dataset from Kaggle: https://kaggle.com #
Ensure 'train.csv' is in your directory #
(df = pd.read_csv('train.csv

HANDLE MISSING VALUES .2 #
Drop columns with excessive missing data or irrelevant features #
(df.drop(['Cabin', 'Name', 'Ticket', 'PassengerId'], axis=1, inplace=True

Impute numerical data (Age) with the median #
(num_imputer = SimpleImputer(strategy='median
(df[['Age']] = num_imputer.fit_transform(df[['Age

(Impute categorical data (Embarked) with the most frequent value (mode #
(cat_imputer = SimpleImputer(strategy='most_frequent
(df[['Embarked']] = cat_imputer.fit_transform(df[['Embarked

Convert categorical text to numerical dummy variables #
(df = pd.get_dummies(df, columns=['Sex', 'Embarked'], drop_first=True

Split data into training and testing sets #
(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

APPLY ML MODELS WITH GRID SEARCH .3 #
    } = param_grid
    ,[n_estimators': [50, 100, 200'
    ,[max_depth': [None, 10, 20'
    [min_samples_split': [2, 5, 10'
    {
grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid,
    (cv=5
    (grid_search.fit(X_train, y_train

("--- print(f"--- Step 3: Best Grid Search Params
    (_print(grid_search.best_params

```

```

FEATURE SELECTION METHODS .4 #
    Method A: Mutual Information #
(mi_selector = SelectKBest(score_func=mutual_info_classif, k=5
    (mi_selector.fit(X_train, y_train
[()]mi_features = X.columns[mi_selector.get_support

    (Method B: Recursive Feature Elimination (RFE #
rfe_selector = RFE(estimator=LogisticRegression(max_iter=1000),
    (n_features_to_select=5
    (rfe_selector.fit(X_train, y_train
[()]rfe_features = X.columns[rfe_selector.get_support

    ("--- print(f"\n--- Step 4: Important Features
    ("{(print(f"Mutual Info Selection: {list(mi_features
    ("{(print(f"RFE Selection: {list(rfe_features

EVALUATE MODELS ON SELECTED FEATURES .5 #
Re-training the best model using only the Mutual Information selected features #
    [X_train_mi = X_train[mi_features
    [X_test_mi = X_test[mi_features

    _best_model = grid_search.best_estimator
    (best_model.fit(X_train_mi, y_train
    (y_pred = best_model.predict(X_test_mi

    ("--- (print(f"\n--- Step 5: Model Performance (Selected Features
    ((print(classification_report(y_test, y_pred

```