

```

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.metrics import precision_score

# Assuming you have your data in a pandas DataFrame called 'df'
# and your target variable is in a column named 'target'

# **Load your data into a pandas DataFrame called 'df' here.**
# **For example, if your data is in a CSV file named 'data.csv', you would use:**
X_train = pd.read_csv('/X_Train_Data_Input.csv')
Y_train = pd.read_csv('/Y_Train_Data_Target.csv')
X_test = pd.read_csv('/X_Test_Data_Input.csv')
Y_test = pd.read_csv('/Y_Test_Data_Target.csv')

# **Create a DataFrame 'df' by merging X_train and Y_train**
df = pd.merge(X_train, Y_train, on='ID', how='left') # Assuming 'ID' is the common column

# First, create X and y from your DataFrame
X = df.drop('target', axis=1) # Features
y = df['target'] # Target variable

# Then split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Now, continue with the rest of your code:

# Load the data
#X = X_train # This line is no longer needed
#y = Y_train.iloc[:len(X_train), 1] # This line is no longer needed

# Split data into training and testing sets
X_train_split, X_test_split, y_train_split, y_test_split = train_test_split(X_train, y_train, test_size=0.2, random_state=42)
# Drop the 'ID' column before imputation
X_train_split = X_train_split.drop(columns=['ID'])
X_test_split = X_test_split.drop(columns=['ID'])
# Handle missing values with imputation
imputer = SimpleImputer(strategy='mean')
X_train_split = imputer.fit_transform(X_train_split)
X_test_split = imputer.transform(X_test_split)


# Create and train the Random Forest model
model = RandomForestClassifier(random_state=42)
model.fit(X_train_split, y_train_split)

# Make predictions
y_train_pred = model.predict(X_train_split)
y_test_pred = model.predict(X_test_split)

# Calculate precision
training_precision = precision_score(y_train_split, y_train_pred)
testing_precision = precision_score(y_test_split, y_test_pred)

# Print the precision
print("Training Precision:", training_precision)
print("Testing Precision:", testing_precision)

```

 Training Precision: 0.9840769648148922
 Testing Precision: 0.8470496894409938

```

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.metrics import recall_score

# Load the datasets
X_train = pd.read_csv('/X_Train_Data_Input.csv')
Y_train = pd.read_csv('/Y_Train_Data_Target.csv')
X_test = pd.read_csv('/X_Test_Data_Input.csv')
Y_test = pd.read_csv('/Y_Test_Data_Target.csv')

# Merge X_train and Y_train to create a complete training dataset
df = pd.merge(X_train, Y_train, on='ID', how='left')

# Extract features (X) and target (y) from the training dataset
X = df.drop('target', axis=1)

```

```

y = df['target']

# Split the training data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Drop the 'ID' column from training and validation sets
X_train = X_train.drop(columns=['ID'])
X_val = X_val.drop(columns=['ID'])

# Handle missing values using imputation
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
X_val = imputer.transform(X_val)

# Create and train the Random Forest model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Make predictions on the validation set
y_val_pred = model.predict(X_val)

# Calculate recall on the validation set
validation_recall = recall_score(y_val, y_val_pred)

# Preprocess the test data (similar to validation data preprocessing)
X_test = X_test.drop(columns=['ID'])
X_test = imputer.transform(X_test) # Use the same imputer fitted on training data

# Make predictions on the test set
y_test_pred = model.predict(X_test)

# Calculate recall on the test set
testing_recall = recall_score(Y_test['target'], y_test_pred)

# Print the recall scores
print("Validation Recall:", validation_recall)
print("Testing Recall:", testing_recall)

```

↔ Validation Recall: 0.9162825379609545
Testing Recall: 0.9172542345408866

```

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.metrics import f1_score

# Load the datasets
X_train = pd.read_csv('/X_Train_Data_Input.csv')
Y_train = pd.read_csv('/Y_Train_Data_Target.csv')
X_test = pd.read_csv('/X_Test_Data_Input.csv')
Y_test = pd.read_csv('/Y_Test_Data_Target.csv')

# Merge X_train and Y_train to create a complete training dataset
df = pd.merge(X_train, Y_train, on='ID', how='left')

# Extract features (X) and target (y) from the training dataset
X = df.drop('target', axis=1)
y = df['target']

# Split the training data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Drop the 'ID' column from training and validation sets
X_train = X_train.drop(columns=['ID'])
X_val = X_val.drop(columns=['ID'])

# Handle missing values using imputation
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
X_val = imputer.transform(X_val)

# Create and train the Random Forest model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Make predictions on the validation set
y_val_pred = model.predict(X_val)

# Calculate F1-score on the validation set
validation_f1 = f1_score(y_val, y_val_pred)

```

```
# Preprocess the test data (similar to validation data preprocessing)
X_test = X_test.drop(columns=['ID'])
X_test = imputer.transform(X_test) # Use the same imputer fitted on training data

# Make predictions on the test set
y_test_pred = model.predict(X_test)

# Calculate F1-score on the test set
testing_f1 = f1_score(Y_test['target'], y_test_pred)

# Print the F1-scores
print("Validation F1-score:", validation_f1)
print("Testing F1-score:", testing_f1)
```

↗ Validation F1-score: 0.8803855798352167
Testing F1-score: 0.8809153175591532

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.metrics import roc_auc_score

# Load the datasets
X_train = pd.read_csv('/X_Train_Data_Input.csv')
Y_train = pd.read_csv('/Y_Train_Data_Target.csv')
X_test = pd.read_csv('/X_Test_Data_Input.csv')
Y_test = pd.read_csv('/Y_Test_Data_Target.csv')

# Merge X_train and Y_train to create a complete training dataset
df = pd.merge(X_train, Y_train, on='ID', how='left')

# Extract features (X) and target (y) from the training dataset
X = df.drop('target', axis=1)
y = df['target']

# Split the training data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Drop the 'ID' column from training and validation sets
X_train = X_train.drop(columns=['ID'])
X_val = X_val.drop(columns=['ID'])

# Handle missing values using imputation
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
X_val = imputer.transform(X_val)

# Create and train the Random Forest model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Make predictions on the validation set (get probabilities for ROC AUC)
y_val_prob = model.predict_proba(X_val)[: , 1] # Probability of positive class

# Calculate ROC AUC on the validation set
validation_roc_auc = roc_auc_score(y_val, y_val_prob)

# Preprocess the test data (similar to validation data preprocessing)
X_test = X_test.drop(columns=['ID'])
X_test = imputer.transform(X_test) # Use the same imputer fitted on training data

# Make predictions on the test set (get probabilities for ROC AUC)
y_test_prob = model.predict_proba(X_test)[: , 1] # Probability of positive class

# Calculate ROC AUC on the test set
testing_roc_auc = roc_auc_score(Y_test['target'], y_test_prob)

# Print the ROC AUC scores
print("Validation ROC AUC:", validation_roc_auc)
print("Testing ROC AUC:", testing_roc_auc)
```

↗ Validation ROC AUC: 0.9938390408171979
Testing ROC AUC: 0.994001775069242

