```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score


# Load the datasets
train_df = pd.read_csv("train_u6lujuX_CVtuZ9i.csv")
test_df = pd.read_csv("test_Y3wMUE5_7gLdaTN.csv")


# Drop Loan_ID as it is not a predictive feature
test_ids = test_df['Loan_ID']  # Save Loan_IDs for submission
train_df.drop(columns=['Loan_ID'], inplace=True)
test_df.drop(columns=['Loan_ID'], inplace=True)


# Separate target variable
y = train_df['Loan_Status']
X = train_df.drop(columns=['Loan_Status'])


# Handling missing values (Apply imputer separately to features only)
imputer = SimpleImputer(strategy='most_frequent')
X.iloc[:, :] = imputer.fit_transform(X)
test_df.iloc[:, :] = imputer.transform(test_df)



# Encode categorical variables
label_encoders = {}
for col in X.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])
    test_df[col] = le.transform(test_df[col])  # Apply same encoding
    label_encoders[col] = le  # Save encoders for later use



# Split 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)


# Train a RandomForest model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
# Make predictions on validation set
y_pred = model.predict(X_val)



# Evaluate model accuracy
accuracy = accuracy_score(y_val, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")
```

```
Model Accuracy: 76.42%
```

```
# Predict on test dataset
test_predictions = model.predict(test_df)
```

Start coding or generate with AI.