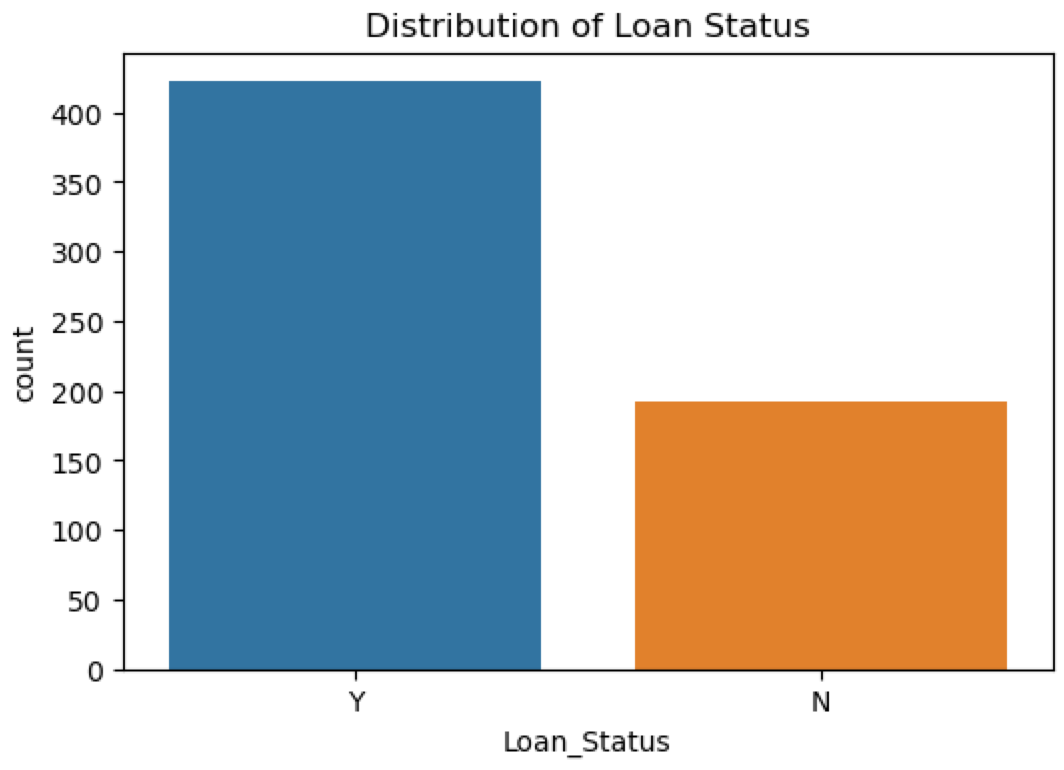
In [1]:

```python
# Importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import joblib

# Load the dataset
link = 'https://github.com/dsrscientist/DSData/raw/master/loan_prediction.csv'
data = pd.read_csv(link)

# EDA Analysis
# Display basic statistics
print(data.describe())

# Check for missing values
print(data.isnull().sum())

# Visualize the distribution of the target variable (Loan_Status)
plt.figure(figsize=(6, 4))
sns.countplot(x='Loan_Status', data=data)
plt.title('Distribution of Loan Status')
plt.show()

# Preprocessing and Feature Engineering
# Handling missing values
data['Gender'].fillna(data['Gender'].mode()[0], inplace=True)
data['Married'].fillna(data['Married'].mode()[0], inplace=True)
data['Dependents'].fillna(data['Dependents'].mode()[0], inplace=True)
data['Self_Employed'].fillna(data['Self_Employed'].mode()[0], inplace=True)
data['LoanAmount'].fillna(data['LoanAmount'].median(), inplace=True)
data['Loan_Amount_Term'].fillna(data['Loan_Amount_Term'].mode()[0], inplace=True)
data['Credit_History'].fillna(data['Credit_History'].mode()[0], inplace=True)

# Encoding categorical variables
le = LabelEncoder()
data['Gender'] = le.fit_transform(data['Gender'])
data['Married'] = le.fit_transform(data['Married'])
data['Dependents'] = le.fit_transform(data['Dependents'])
data['Education'] = le.fit_transform(data['Education'])
data['Self_Employed'] = le.fit_transform(data['Self_Employed'])
data['Property_Area'] = le.fit_transform(data['Property_Area'])
data['Loan_Status'] = le.fit_transform(data['Loan_Status'])

# Feature selection
features = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
            'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term', 'Credit_History', 'Property_Area']
target = 'Loan_Status'

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(data[features], data[target], test_size=0.2, random_state=42)

# Build/Test Multiple Models
# Random Forest Classifier
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)

# Check for overfitting/underfitting
train_predictions = rf_model.predict(X_train)
print(f"Train Accuracy: {accuracy_score(y_train, train_predictions)}")
print(f"Test Accuracy: {accuracy_score(y_test, rf_predictions)}")

# Cross-validation
cv_accuracy = cross_val_score(rf_model, data[features], data[target], scoring='accuracy', cv=5)
print(f"Cross-Validation Accuracy: {np.mean(cv_accuracy)}")

# Hyperparameter Tuning
param_grid = {'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20], 'min_samples_split': [2, 5, 10]}
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, scoring='accuracy', cv=5)
grid_search.fit(data[features], data[target])

# Select the Best Model
best_model = grid_search.best_estimator_

# Save the Best Model for Production
joblib.dump(best_model, 'loan_approval_model.pkl')

# Explanation for Model Selection
# The best model is selected based on the highest accuracy obtained during cross-validation.
# This metric provides a measure of how well the model performs in predicting loan approval status.

# Additional Evaluation Metrics
print("Classification Report:\n", classification_report(y_test, rf_predictions))
print("Confusion Matrix:\n", confusion_matrix(y_test, rf_predictions))
```

```
       ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
count       614.000000         614.000000  592.000000         600.00000
mean       5403.459283        1621.245798  146.412162         342.00000
std        6109.041673        2926.248369   85.587325          65.12041
min         150.000000           0.000000    9.000000          12.00000
25%        2877.500000           0.000000  100.000000         360.00000
50%        3812.500000        1188.500000  128.000000         360.00000
75%        5795.000000        2297.250000  168.000000         360.00000
max       81000.000000       41667.000000  700.000000         480.00000

       Credit_History
count      564.000000
mean         0.842199
std          0.364878
min          0.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          1.000000
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```




```
Train Accuracy: 1.0
Test Accuracy: 0.7723577235772358
Cross-Validation Accuracy: 0.7882713581234173
Classification Report:
               precision    recall  f1-score   support

           0       0.86      0.42      0.56        43
           1       0.75      0.96      0.85        80

    accuracy                           0.77       123
   macro avg       0.81      0.69      0.70       123
weighted avg       0.79      0.77      0.75       123

Confusion Matrix:
 [[18 25]
 [ 3 77]]
```

Importing Libraries: You start by importing the necessary Python libraries for data analysis, visualization, and machine learning.

Loading the Dataset: You load a dataset related to loan predictions from a given link using pandas.

Exploratory Data Analysis (EDA): You conduct exploratory data analysis by displaying basic statistics, checking for missing values, and visualizing the distribution of the target variable (Loan_Status).

Preprocessing and Feature Engineering: You handle missing values by imputing them with mode or median values. Categorical variables are encoded using LabelEncoder. Additionally, you select specific features and the target variable for your model.

Train-Test Split: You split the dataset into training and testing sets using the train_test_split function.

Building and Testing Multiple Models: You use a Random Forest Classifier, fit the model, and make predictions. You check for overfitting/underfitting by comparing the accuracy on both training and testing sets. Cross-validation is performed to obtain a more robust evaluation.

Hyperparameter Tuning: You perform a grid search to find the best hyperparameters for the Random Forest model.

Selecting the Best Model: The best model is selected based on the results of the hyperparameter tuning.

Saving the Model: The best model is saved using the joblib library for potential use in production.

Explanation for Model Selection: A brief explanation is provided on how the best model is selected based on cross-validation accuracy.

Additional Evaluation Metrics: You print the classification report and confusion matrix for further evaluation of the model's performance.