# Code Explanation with Matched Code Snippets

## Importing Required Libraries

The script imports essential libraries for data handling, visualization, and machine learning:  
- `pandas`, `numpy`: Data manipulation and numerical computations.  
- `matplotlib.pyplot`, `seaborn`: Data visualization.  
- `imblearn.over\_sampling.RandomOverSampler`: Handles class imbalance.  
- `sklearn.preprocessing.OrdinalEncoder`: Encodes categorical variables.  
- `sklearn.model\_selection.train\_test\_split`: Splits dataset into training and testing sets.  
- `sklearn.ensemble.RandomForestClassifier`: Implements the Random Forest model.  
- `sklearn.metrics`: Evaluates model performance.

Code:

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from imblearn.over\_sampling import RandomOverSampler  
from sklearn.preprocessing import OrdinalEncoder  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

## Loading and Exploring the Dataset

The dataset is loaded from a CSV file. The script then:  
- Displays the first few rows (`df.head()`).  
- Shows dataset information (`df.info()`).  
- Provides summary statistics (`df.describe()`).  
- Checks for missing values (`df.isna().sum()`) and duplicates (`df.duplicated().sum()`).

Code:

df = pd.read\_csv('telecom\_churn.csv')  
  
print(df.head())  
  
print(df.info())  
  
print(df.describe())  
  
print(df.isna().sum())  
  
print(df.duplicated().sum())

## Defining Features and Target Variable

The script separates the independent variables (features) and dependent variable (target):  
- `y` stores the churn status of customers.  
- `X` includes all relevant features except `customer\_id` and `churn`.

Code:

y = df['churn']  
X = df.drop(['customer\_id', 'churn'], axis=1)

## Handling Missing Values

Missing values in the target column (`churn`) are removed using `dropna()`.

Code:

df = df.dropna(subset=['churn'])

## Encoding Categorical Features

Categorical variables are identified and converted into numerical values using `OrdinalEncoder()`.

Code:

categorical\_cols = X.select\_dtypes(include=['object']).columns  
encoder = OrdinalEncoder()  
X[categorical\_cols] = encoder.fit\_transform(X[categorical\_cols])

## Handling Class Imbalance

Since the dataset may have imbalanced classes (more non-churners than churners), the `RandomOverSampler` is used to balance it.

Code:

oversampler = RandomOverSampler(random\_state=42)  
X\_resampled, y\_resampled = oversampler.fit\_resample(X, y)

## Splitting Data into Training and Testing Sets

The dataset is split into 80% training and 20% testing using `train\_test\_split()`.

Code:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 X\_resampled, y\_resampled, test\_size=0.2, random\_state=42)

## Training the Machine Learning Model

A \*\*Random Forest Classifier\*\* with 100 trees is trained using the `fit()` function.

Code:

clf = RandomForestClassifier(n\_estimators=100, random\_state=42)  
clf.fit(X\_train, y\_train)

## Evaluating the Model

The trained model is tested on unseen data. The following metrics are printed:  
- \*\*Accuracy Score\*\*  
- \*\*Confusion Matrix\*\*  
- \*\*Classification Report\*\*

Code:

y\_pred = clf.predict(X\_test)  
  
print("Accuracy:", accuracy\_score(y\_test, y\_pred))  
print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))  
print("Classification Report:\n", classification\_report(y\_test, y\_pred))

## Visualizing Churn Distribution

A bar plot is created to show the distribution of customers who churned vs. those who stayed.

Code:

sns.countplot(x='churn', data=df)  
plt.title("Churn Distribution")  
plt.show()

## Feature Importance

Feature importance is calculated and plotted to show which features contribute most to churn prediction.

Code:

feature\_importances = clf.feature\_importances\_  
features = X.columns  
  
plt.figure(figsize=(10, 6))  
sns.barplot(x=feature\_importances, y=features)  
plt.title("Feature Importances")  
plt.show()