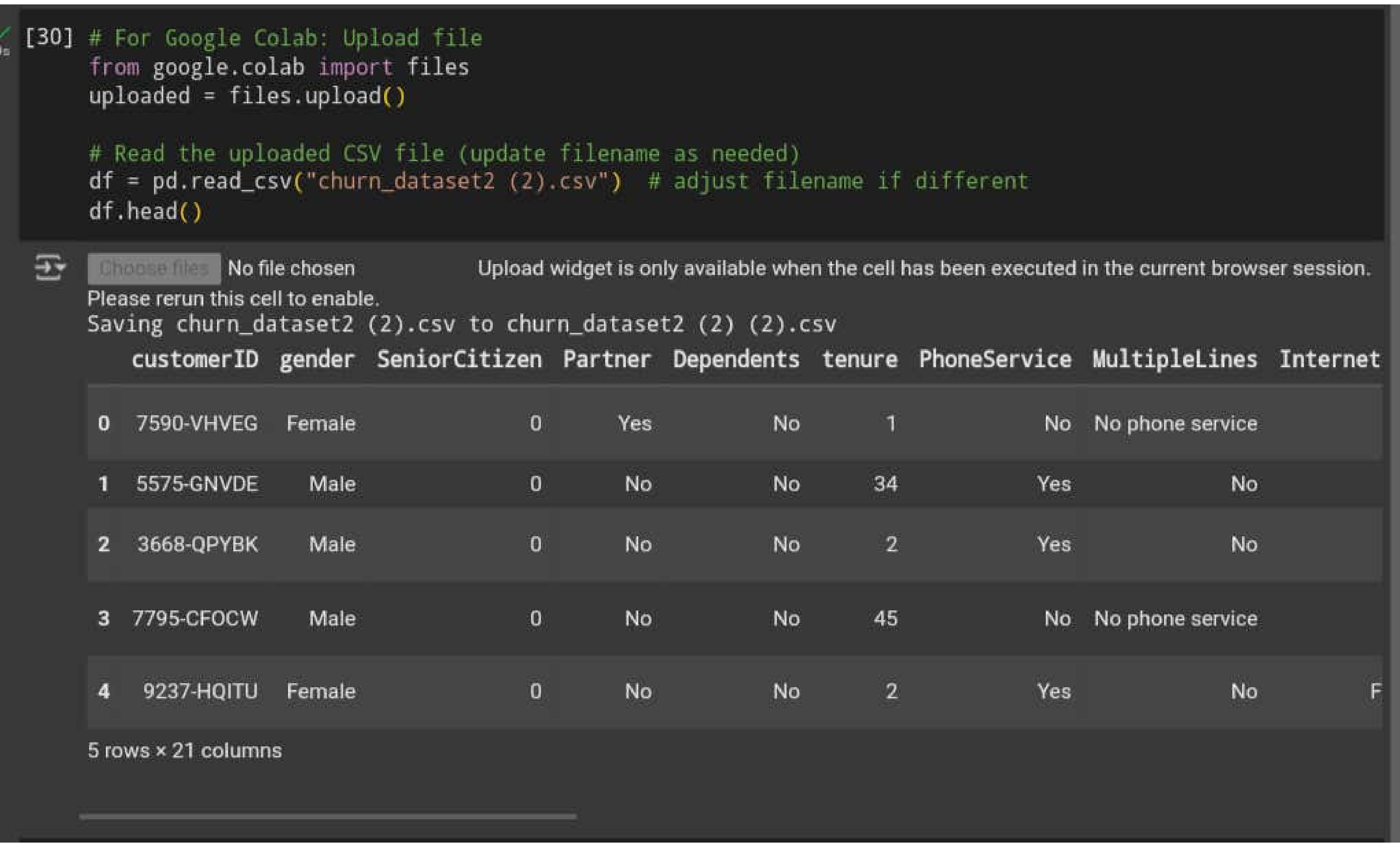


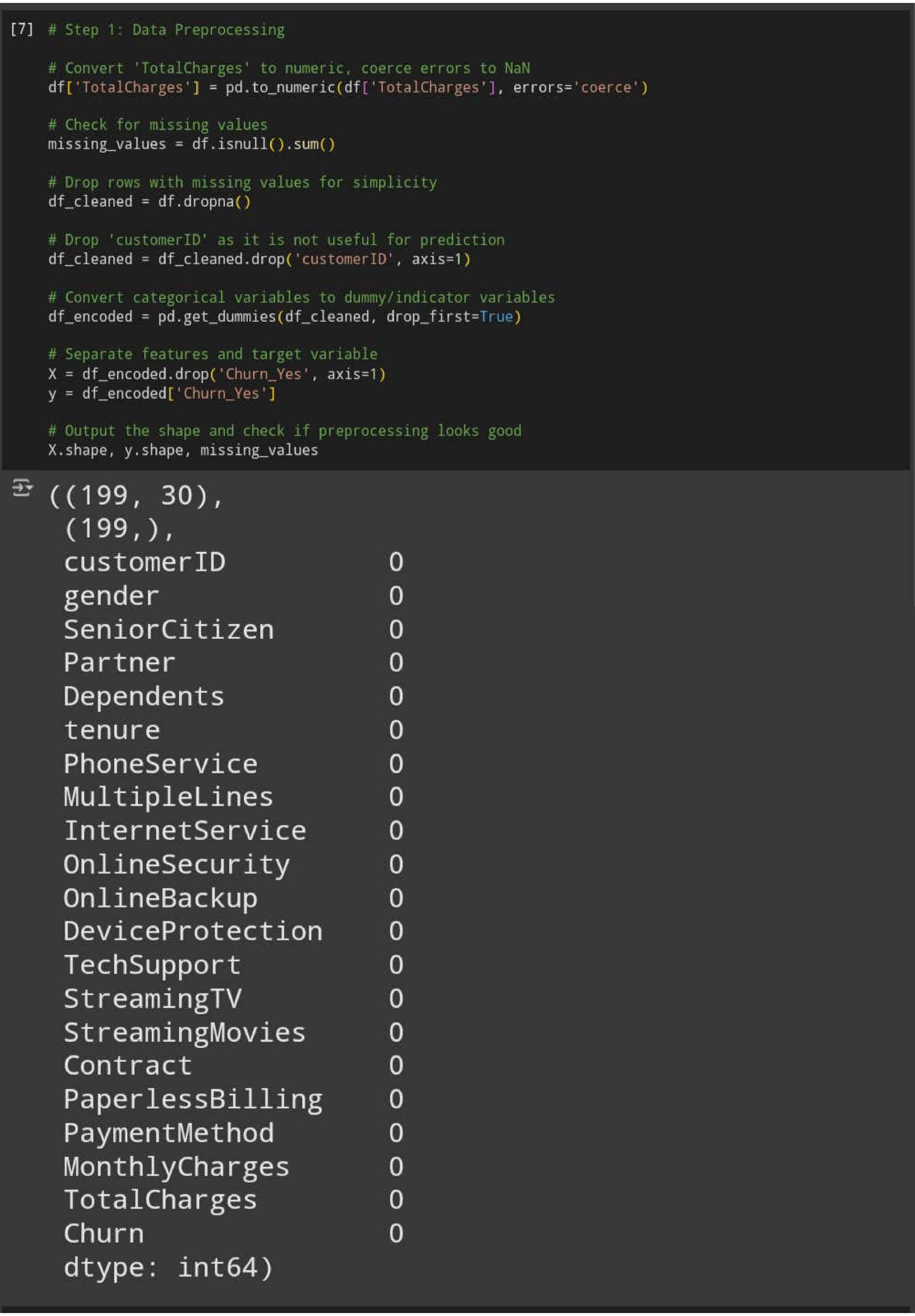
```
[30] # For Google Colab: Upload file
     from google.colab import files
     uploaded = files.upload()

     # Read the uploaded CSV file (update filename as needed)
     df = pd.read_csv("churn_dataset2 (2).csv")  # adjust filename if different
     df.head()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session.
Please rerun this cell to enable.

```
Saving churn_dataset2 (2).csv to churn_dataset2 (2) (2).csv
```

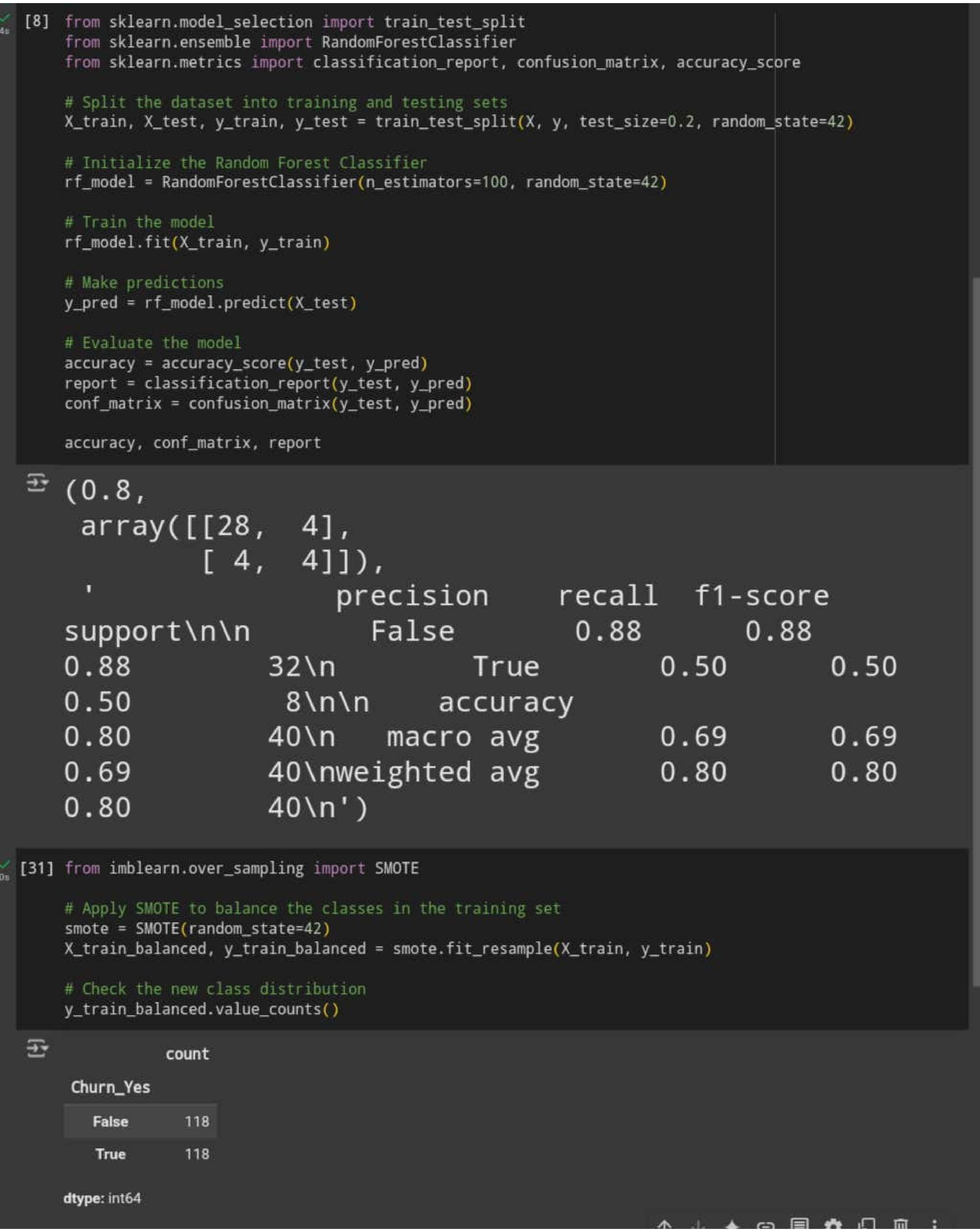
	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Internet
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	
4	9237-HQITU	Female	0	No	No	2	Yes	No	F

5 rows × 21 columns



```
[7] # Step 1: Data Preprocessing

    # Convert 'TotalCharges' to numeric, coerce errors to NaN
    df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

    # Check for missing values
    missing_values = df.isnull().sum()

    # Drop rows with missing values for simplicity
    df_cleaned = df.dropna()

    # Drop 'customerID' as it is not useful for prediction
    df_cleaned = df_cleaned.drop('customerID', axis=1)

    # Convert categorical variables to dummy/indicator variables
    df_encoded = pd.get_dummies(df_cleaned, drop_first=True)

    # Separate features and target variable
    X = df_encoded.drop('Churn_Yes', axis=1)
    y = df_encoded['Churn_Yes']

    # Output the shape and check if preprocessing looks good
    X.shape, y.shape, missing_values
```

```
((199, 30),
 (199,),
 customerID          0
 gender              0
 SeniorCitizen       0
 Partner             0
 Dependents          0
 tenure              0
 PhoneService        0
 MultipleLines       0
 InternetService     0
 OnlineSecurity      0
 OnlineBackup        0
 DeviceProtection    0
 TechSupport         0
 StreamingTV         0
 StreamingMovies     0
 Contract            0
 PaperlessBilling    0
 PaymentMethod       0
 MonthlyCharges      0
 TotalCharges        0
 Churn               0
 dtype: int64)
```



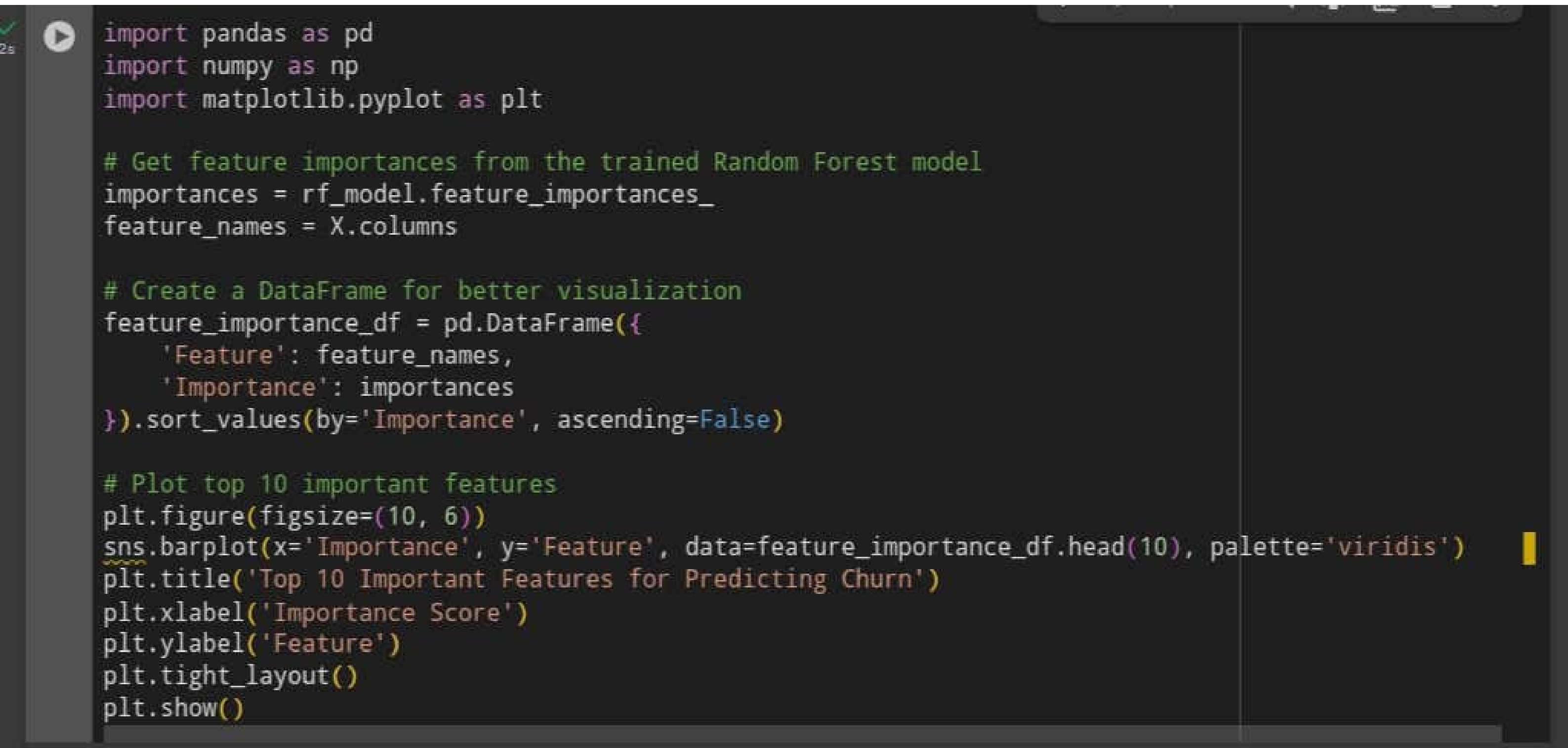
```
[8] from sklearn.model_selection import train_test_split
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

    # Split the dataset 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)

    # Initialize the Random Forest Classifier
    rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

    # Train the model
    rf_model.fit(X_train, y_train)

    # Make predictions
    y_pred = rf_model.predict(X_test)

    # Evaluate the model
    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)
    conf_matrix = confusion_matrix(y_test, y_pred)

    accuracy, conf_matrix, report
```

```
(0.8,
 array([[28,  4],
        [ 4,  4]]),
 '              precision    recall  f1-score
support\n\n       False       0.88      0.88
0.88        32\n        True       0.50      0.50
0.50         8\n\n    accuracy
0.80        40\n   macro avg       0.69      0.69
0.69        40\nweighted avg       0.80      0.80
0.80        40\n')
```

```
[31] from imblearn.over_sampling import SMOTE

     # Apply SMOTE to balance the classes in the training set
     smote = SMOTE(random_state=42)
     X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)

     # Check the new class distribution
     y_train_balanced.value_counts()
```

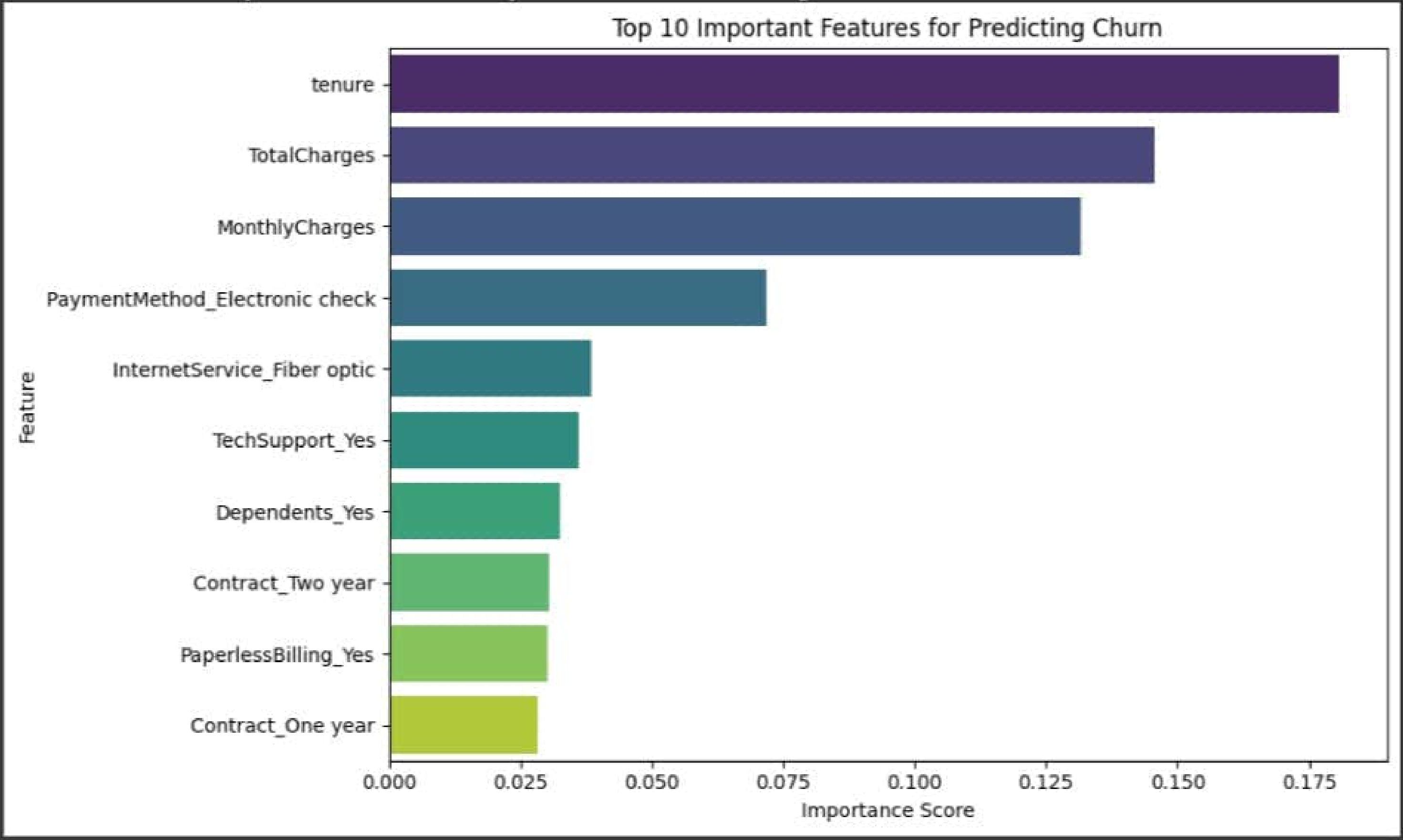
	count
Churn_Yes	
False	118
True	118

dtype: int64



```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Get feature importances from the trained Random Forest model
importances = rf_model.feature_importances_
feature_names = X.columns

# Create a DataFrame for better visualization
feature_importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

# Plot top 10 important features
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df.head(10), palette='viridis')
plt.title('Top 10 Important Features for Predicting Churn')
plt.xlabel('Importance Score')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()
```

```
<ipython-input-36-9cd4e63fee76>:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecat

  sns.barplot(x='Importance', y='Feature', data=featu
```


Top 10 Important Features for Predicting Churn
tenure
TotalCharges
MonthlyCharges
PaymentMethod_Electronic check
InternetService_Fiber optic
TechSupport_Yes
Dependents_Yes
Contract_Two year
PaperlessBilling_Yes
Contract_One year
Feature
Importance Score