```
# IMPORTANT: SOME KAGGLE DATA SOURCES ARE PRIVATE
# RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES.
import kagglehub
kagglehub.login()



# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

sajansinghshergill_forage_chips_customer_analysis_plan_task_4_path = kagglehub.dataset_download('sajansinghshergill/forage-c

print('Data source import complete.')



# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a versi
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session


import pandas as pd

# Load the dataset for predictions
data = pd.read_csv('/content/data_for_predictions.csv')

# Display the first few rows to understand its structure
print(data.head())
```

```
   forecast_price_energy_peak  ...  months_modif_prod  months_renewal  \
0                    0.098142  ...                  2               6
1                    0.000000  ...                 76               4
2                    0.087899  ...                 68               8
3                    0.000000  ...                 69               9
4                    0.100015  ...                 71               9

   channel_MISSING  channel_ewpakwlliwisiwduibdlfmalxowmwpci  \
0                0                                         0
1                1                                         0
2                0                                         0
3                0                                         0
4                1                                         0

   channel_foosdfpfkusacimwkcsosbicdxkicaua  \
0                                         1
1                                         0
2                                         1
```

```
3                                          1
4                                          1

   origin_up_ldkssxwpmemidmecebumciepifcamkci  \
0                                          0
1                                          0
2                                          0
3                                          0
4                                          0

   origin_up_lxidpiddsbxsbosboudacockeimpuepw
0                                          1
1                                          0
2                                          0
3                                          0
4                                          0

[5 rows x 64 columns]
```

```
# Assuming 'churn' is the target variable
X = data.drop('churn', axis=1)  # Features
y = data['churn']               # Target variable
```

```
from sklearn.model_selection import train_test_split

# 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)
```

```
# Check for categorical columns
categorical_cols = X.select_dtypes(include=['object']).columns.tolist()
print("Categorical Columns:", categorical_cols)
```

```
Categorical Columns: ['id']
```

```
print("Shape of the new feature set:", X_encoded.shape)
```

```
Shape of the new feature set: (14606, 14667)
```

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

# Split the encoded dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# Create and train the model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions
y_pred = rf_model.predict(X_test)

# Evaluate the model
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
[[2616    1]
 [ 296    9]]

Classification Report:
              precision    recall  f1-score   support

           0       0.90      1.00      0.95      2617
           1       0.90      0.03      0.06       305

    accuracy                           0.90      2922
   macro avg       0.90      0.51      0.50      2922
weighted avg       0.90      0.90      0.85      2922

Accuracy Score: 0.8983572895277208
```

```
# Apply one-hot encoding
X_encoded = pd.get_dummies(X, columns=categorical_cols, drop_first=True)
```

```
from sklearn.ensemble import RandomForestClassifier

# Create the model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

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

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