**Goal: Churn Prediction with Balanced Classes Using SMOTE**

**🔹 1. Imports and Setup**

python

CopierModifier

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

This imports the necessary libraries to:

* Handle data (pandas, numpy)
* Split data (train\_test\_split)
* Scale features (StandardScaler)
* Train model (LogisticRegression)
* Evaluate performance (accuracy\_score, etc.)

**🔹 2. One-hot Encoding**

python

CopierModifier

categorical\_columns = ['INDUSTRY']

data\_encoded = pd.get\_dummies(data, columns=categorical\_columns, drop\_first=True)

* Converts the INDUSTRY categorical column into binary (dummy) variables.
* drop\_first=True avoids multicollinearity by removing the first dummy column.

**🔹 3. Prepare Features and Target**

python

CopierModifier

X = data\_encoded.drop(['CUSTOMER\_ID', 'CHURN\_LABEL', 'CUSTOMER\_STATUS'], axis=1)

y = data\_encoded['CHURN\_LABEL']

* X: all features (input variables).
* y: the target (what we're predicting = churn label).
* Removes IDs and irrelevant labels.

**🔹 4. Train/Test Split**

python

CopierModifier

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

* 80% training / 20% testing split.
* random\_state=42 ensures reproducibility.

**🔹 5. Scaling Numerical Features**

python

CopierModifier

scaler = StandardScaler()

numerical\_columns = ['MRR', 'SUBSCRIPTION\_DURATION', 'AVG\_MONTHLY\_LOGINS', 'AVG\_TICKETS\_RAISED', 'AVG\_FEATURE\_USAGE']

X\_train\_df = pd.DataFrame(X\_train, columns=X.columns)

X\_test\_df = pd.DataFrame(X\_test, columns=X.columns)

X\_train\_df[numerical\_columns] = scaler.fit\_transform(X\_train\_df[numerical\_columns])

X\_test\_df[numerical\_columns] = scaler.transform(X\_test\_df[numerical\_columns])

* Scales **only numerical columns** to standardize them (mean=0, std=1).
* Helps logistic regression converge faster and more accurately.

**🔹 6. Handling Class Imbalance: SMOTE**

python

CopierModifier

from imblearn.over\_sampling import SMOTE

smote = SMOTE(random\_state=42)

X\_train\_smote, y\_train\_smote = smote.fit\_resample(X\_train\_df, y\_train)

* **SMOTE (Synthetic Minority Oversampling Technique)** creates synthetic samples for the minority class (e.g. churners) in the training set.
* Helps the model learn **balanced patterns** instead of being biased toward the majority class.

**🔹 7. Train the Model**

python

CopierModifier

model = LogisticRegression(random\_state=42)

model.fit(X\_train\_smote, y\_train\_smote)

* Trains logistic regression on **balanced** training data (after SMOTE).

**🔹 8. Evaluate the Model**

python

CopierModifier

y\_pred = model.predict(X\_test\_df)

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

* Tests the model on the **original** (imbalanced) test data.
* Shows performance metrics like:
  + **Accuracy**
  + **Precision/Recall/F1-score**
  + **Confusion Matrix**

**✅ Final Result:**

* You achieved **~98.5% accuracy**, likely due to:
  + SMOTE balancing
  + Good features
  + Proper scaling
* The model seems to perform well on both churners and non-churners.

**Let Me Know If You Want To:**

* Visualize results (ROC curve, feature importance, etc.)
* Try a different model (e.g. Random Forest, XGBoost)
* Automate preprocessing in a pipeline (with Pipeline or ColumnTransformer)

You're on the right track!

Splitting into training and testing datasets

Code

"# Split the dataset into training and testing sets\n",

"from sklearn.model\_selection import train\_test\_split\n",

"\n",

"X = data.drop('CHURN\_LABEL', axis=1) # Features\n",

"y = data['CHURN\_LABEL'] # Target variable\n",

"\n",

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

]

},

{

"cell\_type": "code",

"execution\_count": 32,

"id": "10449058-f585-4a46-a621-8aa63aebc3d7",

"metadata": {},

"outputs": [],

"source": [

"import pandas as pd\n",

"import numpy as np\n",

"from sklearn.model\_selection import train\_test\_split\n",

"from sklearn.preprocessing import StandardScaler, LabelEncoder\n",

"from sklearn.linear\_model import LogisticRegression\n",

"from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix"

]

},

{

"cell\_type": "code",

"execution\_count": 45,

"id": "182f10b4-f25b-43df-be03-35eeefb59bc5",

"metadata": {},

"outputs": [],

"source": [

"# 1. Encode remaining categorical columns\n",

"categorical\_columns = ['INDUSTRY'] # Only include columns that haven't been encoded yet\n",

"data\_encoded = pd.get\_dummies(data, columns=categorical\_columns, drop\_first=True)"

]

},

{

"cell\_type": "code",

"execution\_count": 46,

"id": "95d6ec5d-61c7-4930-aa81-7e2a3b44e177",

"metadata": {},

"outputs": [],

"source": [

"# 2. Prepare features (X) and target (y)\n",

"# Remove CUSTOMER\_ID, CHURN\_LABEL, and CUSTOMER\_STATUS\_Churned from features\n",

"X = data\_encoded.drop(['CUSTOMER\_ID', 'CHURN\_LABEL', 'CUSTOMER\_STATUS'], axis=1)\n",

"y = data\_encoded['CHURN\_LABEL']"

]

},

{

"cell\_type": "code",

"execution\_count": 47,

"id": "b63ceaef-fd55-4abe-8d2e-dad6ede10678",

"metadata": {},

"outputs": [],

"source": [

"# 3. Split the data\n",

"from sklearn.model\_selection import train\_test\_split\n",

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

]

},

{

"cell\_type": "code",

"execution\_count": 48,

"id": "4eead8bc-6f7f-400d-98b3-7fa7034481f8",

"metadata": {},

"outputs": [],

"source": [

"# 4. Scale the numerical features\n",

"from sklearn.preprocessing import StandardScaler\n",

"scaler = StandardScaler()\n",

"numerical\_columns = ['MRR', 'SUBSCRIPTION\_DURATION', 'AVG\_MONTHLY\_LOGINS',\n",

" 'AVG\_TICKETS\_RAISED', 'AVG\_FEATURE\_USAGE']"

]

},

{

"cell\_type": "code",

"execution\_count": 49,

"id": "c6a11496-770e-4860-a1a3-bcfa2dbfdd44",

"metadata": {},

"outputs": [],

"source": [

"# Create a copy of X\_train and X\_test as pandas DataFrames\n",

"X\_train\_df = pd.DataFrame(X\_train, columns=X.columns)\n",

"X\_test\_df = pd.DataFrame(X\_test, columns=X.columns)\n",

"\n",

"# Scale only numerical columns\n",

"X\_train\_df[numerical\_columns] = scaler.fit\_transform(X\_train\_df[numerical\_columns])\n",

"X\_test\_df[numerical\_columns] = scaler.transform(X\_test\_df[numerical\_columns])"

]

},

{

"cell\_type": "code",

"execution\_count": 50,

"id": "758ee89c-a50b-4600-b5f7-047b2e7a66d2",

"metadata": {},

"outputs": [],

"source": [

"# 5. Train and evaluate the model\n",

"from sklearn.linear\_model import LogisticRegression\n",

"from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix\n",

"\n",

"model = LogisticRegression(random\_state=42)\n",

"model.fit(X\_train\_df, y\_train)\n",

"\n",

"# Make predictions\n",

"y\_pred = model.predict(X\_test\_df)"

]

},

{

"cell\_type": "code",

"execution\_count": 56,

"id": "022fd696-0225-4802-bab4-aa3b50c74a9d",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Collecting imbalanced-learn\n",

" Downloading imbalanced\_learn-0.13.0-py3-none-any.whl.metadata (8.8 kB)\n",

"Requirement already satisfied: numpy<3,>=1.24.3 in c:\\users\\parth badani\\appdata\\local\\programs\\python\\python312\\lib\\site-packages (from imbalanced-learn) (1.26.4)\n",

"Requirement already satisfied: scipy<2,>=1.10.1 in c:\\users\\parth badani\\appdata\\local\\programs\\python\\python312\\lib\\site-packages (from imbalanced-learn) (1.14.1)\n",

"Requirement already satisfied: scikit-learn<2,>=1.3.2 in c:\\users\\parth badani\\appdata\\local\\programs\\python\\python312\\lib\\site-packages (from imbalanced-learn) (1.5.1)\n",

"Collecting sklearn-compat<1,>=0.1 (from imbalanced-learn)\n",

" Downloading sklearn\_compat-0.1.3-py3-none-any.whl.metadata (18 kB)\n",

"Requirement already satisfied: joblib<2,>=1.1.1 in c:\\users\\parth badani\\appdata\\local\\programs\\python\\python312\\lib\\site-packages (from imbalanced-learn) (1.4.2)\n",

"Requirement already satisfied: threadpoolctl<4,>=2.0.0 in c:\\users\\parth badani\\appdata\\local\\programs\\python\\python312\\lib\\site-packages (from imbalanced-learn) (3.5.0)\n",

"Downloading imbalanced\_learn-0.13.0-py3-none-any.whl (238 kB)\n",

"Downloading sklearn\_compat-0.1.3-py3-none-any.whl (18 kB)\n",

"Installing collected packages: sklearn-compat, imbalanced-learn\n",

"Successfully installed imbalanced-learn-0.13.0 sklearn-compat-0.1.3\n",

"Note: you may need to restart the kernel to use updated packages.\n"

]

}

],

"source": [

"pip install imbalanced-learn"

]

},

{

"cell\_type": "code",

"execution\_count": 57,

"id": "2507a66d-1081-4225-b4ad-05bbfcc3f6a1",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Accuracy: 0.985\n",

"\n",

"Classification Report:\n",

" precision recall f1-score support\n",

"\n",

" 0 1.00 0.99 0.99 845\n",

" 1 0.93 0.97 0.95 155\n",

"\n",

" accuracy 0.98 1000\n",

" macro avg 0.96 0.98 0.97 1000\n",

"weighted avg 0.99 0.98 0.99 1000\n",

"\n",

"\n",

"Confusion Matrix:\n",

" [[834 11]\n",

" [ 4 151]]\n"

]

},

{

"name": "stderr",

"output\_type": "stream",

"text": [

"C:\\Users\\Parth Badani\\AppData\\Local\\Programs\\Python\\Python312\\Lib\\site-packages\\sklearn\\linear\_model\\\_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):\n",

"STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.\n",

"\n",

"Increase the number of iterations (max\_iter) or scale the data as shown in:\n",

" https://scikit-learn.org/stable/modules/preprocessing.html\n",

"Please also refer to the documentation for alternative solver options:\n",

" https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression\n",

" n\_iter\_i = \_check\_optimize\_result(\n"

]

}

],

"source": [

"from imblearn.over\_sampling import SMOTE\n",

"from sklearn.model\_selection import train\_test\_split\n",

"from sklearn.linear\_model import LogisticRegression\n",

"from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score\n",

"\n",

"# Split the data into training and testing sets\n",

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

"\n",

"# Apply SMOTE to the training data\n",

"smote = SMOTE(random\_state=42)\n",

"X\_train\_smote, y\_train\_smote = smote.fit\_resample(X\_train, y\_train)\n",

"\n",

"# Train the model on the oversampled data\n",

"model = LogisticRegression(random\_state=42)\n",

"model.fit(X\_train\_smote, y\_train\_smote)\n",

"\n",

"# Make predictions on the test set\n",

"y\_pred = model.predict(X\_test)\n",

"\n",

"# Evaluate the model\n",

"print(\"Accuracy:\", accuracy\_score(y\_test, y\_pred))\n",

"print(\"\\nClassification Report:\\n\", classification\_report(y\_test, y\_pred))\n",

"print(\"\\nConfusion Matrix:\\n\", confusion\_matrix(y\_test, y\_pred))"

]

},

{

"cell\_type": "code",

"execution\_count": 58,

"id": "a00732c6-edee-419a-b710-810a0a89ae9a",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"\n",

"Top 10 Most Important Features:\n",

" Feature Importance\n",

"1 SUBSCRIPTION\_DURATION 12.308308\n",

"2 AVG\_MONTHLY\_LOGINS 3.982063\n",

"7 PLAN\_Growth 3.163870\n",

"8 PLAN\_Pro 1.140580\n",

"9 COUNTRY\_Brazil 1.037213\n",

"25 INDUSTRY\_Retail 0.828120\n",

"26 INDUSTRY\_Technology 0.827398\n",

"24 INDUSTRY\_Real Estate 0.820352\n",

"16 COUNTRY\_UK 0.745403\n",

"19 INDUSTRY\_Healthcare 0.724600\n"

]

},

{

"data": {