Title of the Project

Bank Customer Churn Model using python

Objective

The objective of this project is to predict bank customer churn using machine learning models. By analysing customer data, we aim to identify patterns and factors that contribute to customer attrition. This insight will help the bank implement targeted retention strategies, ultimately enhancing customer satisfaction and reducing churn.

Data Source

The dataset used for this project is available at:
Bank Churn Modelling Dataset

Import 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

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import confusion_matrix, classification_report

from sklearn.model_selection import GridSearchCV

from imblearn.under_sampling import RandomUnderSampler

from imblearn.over_sampling import RandomOverSampler
```

Import Data

```
df = pd.read_csv('https://github.com/YBI-Foundation/Dataset/raw/main/Bank%20Churn%20Modelling.csv')

df.head()
```

Describe Data

```
df.info()

df.describe()

df.duplicated('CustomerId').sum()  # Check for duplicates

df = df.set_index('CustomerId')  # Set CustomerId as the index
```

Data Visualization

Visualize the data to identify patterns and outliers:

```
sns.countplot(x='Churn', data=df)
plt.show()

df['Zero Balance'] = np.where(df['Balance']>0, 1, 0)
df['Zero Balance'].hist()
plt.show()

df.groupby(['Churn','Geography']).count().plot(kind='bar')
plt.show()
```

Data Preprocessing

- **Encoding Categorical Variables**: Convert categorical variables into numerical values.
- **Handle Missing Data**: (If any)
- **Balance the Dataset**: Handle imbalanced data using Random Under Sampling (RUS) and Random Over Sampling (ROS).

```
# Encoding categorical variables
df.replace({'Geography': {'France': 2, 'Germany': 1, 'Spain': 0}}, inplace=True)
df.replace({'Gender': {'Male': 0, 'Female': 1}}, inplace=True)
df.replace({'Num Of Products': {1: 0, 2: 1, 3: 1, 4: 1}}, inplace=True)

# Handle imbalance in the dataset
rus = RandomUnderSampler(random_state=2529)
x_rus, y_rus = rus.fit_resample(df.drop(['Surname','Churn'], axis=1), df['Churn'])

ros = RandomOverSampler(random_state=252)
x_ros, y_ros = ros.fit_resample(df.drop(['Surname','Churn'], axis=1), df['Churn'])
```

Define Target Variable (y) and Feature Variables (X)

```
X = df.drop(['Surname', 'Churn'], axis=1)

y = df['Churn']
```

Train Test Split

```
# Split the original, undersampled, and oversampled data

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=2529)

x_train_rus, x_test_rus, y_train_rus, y_test_rus = train_test_split(x_rus, y_rus, test_size=0.3,
random_state=2529)

x_train_ros, x_test_ros, y_train_ros, y_test_ros = train_test_split(x_ros, y_ros, test_size=0.3,
random_state=2529)
```

Data Standardization

```
sc = StandardScaler()

# Standardize the original data

x_train[['CreditScore', 'Age', 'Tenure', 'Balance', 'Estimated Salary']] =
sc.fit_transform(x_train[['CreditScore', 'Age', 'Tenure', 'Balance', 'Estimated Salary']])

# Standardize the undersampled and oversampled data

x_train_rus[['CreditScore', 'Age', 'Tenure', 'Balance', 'Estimated Salary']] =
sc.fit_transform(x_train_rus[['CreditScore', 'Age', 'Tenure', 'Balance', 'Estimated Salary']])

x_train_ros[['CreditScore', 'Age', 'Tenure', 'Balance', 'Estimated Salary']] =
sc.fit_transform(x_train_ros[['CreditScore', 'Age', 'Tenure', 'Balance', 'Estimated Salary']])
```

Modeling

Use Support Vector Machine (SVM) for prediction:

```
svc = SVC()

svc.fit(x_train, y_train)

y_pred = svc.predict(x_test)
```

Model Evaluation

```
confusion_matrix(y_test, y_pred)
print(classification_report(y_test, y_pred))
```

Hyperparameter Tuning

```
param_grid = {'C': [0.1, 1, 10], 'gamma': [1, 0.1, 0.01], 'kernel': ['rbf'], 'class_weight': ['balanced']}
grid = GridSearchCV(SVC(), param_grid, refit=True, verbose=2, cv=2)
grid.fit(x_train, y_train)
grid_predictions = grid.predict(x_test)

confusion_matrix(y_test, grid_predictions)
print(classification_report(y_test, grid_predictions))
```

Prediction with Random Under Sampling and Over Sampling

Train and evaluate models on undersampled and oversampled data:

```
# Model with random under sampling
svc_rus = SVC()
svc_rus.fit(x_train_rus, y_train_rus)
y_pred_rus = svc_rus.predict(x_test_rus)
print(classification_report(y_test_rus, y_pred_rus))

# Model with random over sampling
svc_ros = SVC()
svc_ros.fit(x_train_ros, y_train_ros)
y_pred_ros = svc_ros.predict(x_test_ros)
print(classification_report(y_test_ros, y_pred_ros))
```

Explanation

In this project, we built a predictive model to identify customers who are likely to churn. By using various sampling techniques and machine learning algorithms, we successfully trained models that help predict churn, allowing the bank to take proactive measures in retaining customers. We evaluated model performance using metrics like accuracy, precision, recall, and F1-score, and improved results with hyperparameter tuning.