Customer Churn Prediction

This project involves building a machine learning model to predict customer churn using historical customer data. I have experimented with three algorithms: Logistic Regression and Random Forests to classify customers as churned or retained based on their behavior and attributes.

Steps:

1. Import necessary packages

Import all the required libraries and modules such as pandas for data handling, numpy for numerical operations, scikit-learn modules for machine learning tasks like model selection, preprocessing, metrics, and algorithms, seaborn and matplotlib for visualization, and joblib for model persistence.

2. Load the dataset

Read the CSV file 'customer-Churn.csv' into a pandas DataFrame df.

3. Data exploration

print(df.head()): Display the first few rows of the dataset to understand its structure.
print(df.info()): Print concise summary of the DataFrame to get an overview of columns, data types, and missing values.

4. Prepare data for modeling

Define features and target: Assume the last column as the target variable (y) and the rest as features (X).
Encode the target variable: Use LabelEncoder to convert categorical target variable (y) to numerical values.
Handle missing values: If there are any missing values in features (X), fill them with the mean of the column.
One-hot encode categorical variables: Convert categorical variables in X to dummy/indicator variables.

5. Scale the features

Standardize the features (X) to have zero mean and unit variance using StandardScaler.

6. Split the dataset

Split the data into training and testing sets (80% training, 20% testing) using train_test_split.

7. Define evaluation function

evaluate_model function is defined to evaluate the model performance using metrics such as accuracy, recall, precision, F1 score, and display a confusion matrix and ROC curve if applicable.

8. Model training and evaluation

Logistic Regression: Initialize a LogisticRegression model, fit it to the training data (X_train, y_train), and evaluate its performance using evaluate_model.
Random Forest: Initialize a RandomForestClassifier, fit it to the training data, and evaluate its performance similarly.

9. Comparison of models

Store the trained models (log_reg and random_forest) in a dictionary models.
Evaluate each model's performance on the test set and store metrics (accuracy, recall, precision, F1 score) in results.
Create a DataFrame results_df to compare and print the performance metrics of both models.
Identify the best model based on the highest F1 score from results_df.

10. Save the best model

Save the best performing model (best_model) using joblib.dump to a file named 'best_model_churn.pkl'.

11. Load and verify the model

Load the saved model (best_model_churn.pkl) using joblib.load and make predictions (y_pred_loaded) on the test set to verify its functionality.