Build a Logistics Regression model for Customer Churn

Business Objective

Predicting a qualitative response for observation can be referred to as classifying that observation since it involves assigning the observation to a category or class. Classification forms the basis for Logistic Regression. Logistic Regression is a supervised algorithm used to predict a dependent variable that is categorical or discrete. Logistic regression models the data using the sigmoid function.

Churned Customers are those who have decided to end their relationship with their existing company. In our case study, we will be working on a churn dataset.

XYZ is a service-providing company that provides customers with a one-year subscription plan for their product. The company wants to know if the customers will renew the subscription for the coming year or not.

Data Description

The CSV consists of around 2000 rows and 16 columns

Features:

1. Year
2. Customer_id - unique id
3. Phone_no - customer phone no
4. Gender -Male/Female
5. Age
6. No of days subscribed - the number of days since the subscription
7. Multi-screen - does the customer have a single/ multiple screen subscription
8. Mail subscription - customer receive mails or not
9. Weekly mins watched - number of minutes watched weekly
10. Minimum daily mins - minimum minutes watched
11. Maximum daily mins - maximum minutes watched
12. Weekly nights max mins - number of minutes watched at night time
13. Videos watched - total number of videos watched
14. Maximum_days_inactive - days since inactive
15. Customer support calls - number of customer support calls
16. Churn -
    - 1- Yes
    - 0 - No

Aim

Build a logistics regression learning model on the given dataset to determine whether the customer will churn or not.

Tech stack

- Language - Python
- Libraries - numpy, pandas, matplotlib, seaborn, sklearn, pickle, imblearn, statsmodel

Approach

1. Importing the required libraries and reading the dataset.
2. Inspecting and cleaning up the data
3. Perform data encoding on categorical variables
4. Exploratory Data Analysis (EDA)
    - Data Visualization
5. Feature Engineering
    - Dropping of unwanted columns
6. Model Building
    - Using the statsmodel library
7. Model Building
    - Performing train test split
    - Logistic Regression Model
8. Model Validation (predictions)
    - Accuracy score
    - Confusion matrix
    - ROC and AUC
    - Recall score
    - Precision score
    - F1-score
9. Handling the unbalanced data
    - With balanced weights
    - Random weights
    - Adjusting imbalanced data
    - Using SMOTE
10. Feature Selection
    - Barrier threshold selection
    - RFE method
11. Save the model in the form of a pickle file.

Modular code overview

```
input
  |_data_regression.csv

src
  |_Engine.py
  |_ML_Pipeline
            |_encoding.py
            |_evaluate_metrics.py
            |_feature_engg.py
            |_imbalanced_data.py
            |_ml_model.py
            |_stats_model.py
            |_rescale_variables.py
            |_scaler.py
            |_train_model.py
            |_utils.py

lib
  |_[DEMO]_logistic_regression_E2E.ipynb

output
  |_adjusted_model.pkl
  |_balanced1_model.pkl
  |_balanced2_model.pkl
  |_Log_ROC.png
  |_model_rfe_feat.pkl
  |_model_stats.pkl
  |_model_var_feat.pkl
  |_model1.pkl
  |_smote_model.pkl
```

Once you unzip the modular_code.zip file you can find the following folders within it.

1. input
2. src
3. output
4. lib

1. Input folder **-** It contains all the data that we have for analysis. There is one csv file in our case,
    - Data_regression

2. Src folder - This is the most important folder of the project. This folder contains all the modularized code for all the above steps in a modularized manner. This folder consists of:

- Engine.py
- ML_Pipeline
  The ML_pipeline is a folder that contains all the functions put into different python files, which are appropriately named. These python functions are then called inside the engine.py file.

3. Output folder **–** The output folder contains the best-fitted models that we trained for this data. These models can be easily loaded and used for future use and the user need not have to train all the models from the beginning.
   Note: This model is built over a chunk of data. One can obtain the model for the entire data by running engine.py by taking the entire data to train the models.

4. Lib folder **-** This is a reference folder. It contains the original ipython notebook that we saw in the videos.

Project Takeaways

1. Understanding the basics of classification
2. Introduction to Logistics regression
3. Understanding the logit function
4. Coefficients in logistics regression
5. Concept of maximum log-likelihood
6. Performance metrics like confusion metric, recall, accuracy, precision, f1-score, AUC, and ROC
7. Importing the dataset and required libraries.
8. Performing basic Exploratory Data Analysis (EDA).
9. Using python libraries such as matplotlib and seaborn for data interpretation and advanced visualizations.
10. Data inspection and cleaning
11. Using statsmodel and sklearn libraries to build the model
12. Splitting Dataset into Train and Test using sklearn.
13. Training a model using Classification techniques like Logistics Regression,
14. Making predictions using the trained model.
15. Gaining confidence in the model using metrics such as accuracy score, confusion matrix, recall, precision, and f1 score
16. Handling the unbalanced data using various methods.
17. Performing feature selection with multiple methods
18. Saving the best model in pickle format for future use.