# Customer Churn Prediction

Data science graduation project


Our Team

Mohammed Alkarmi

Thamer Smadi

Mohammed Sharap

## Table of contents

# 1. Introduction

The dataset you'll be using to develop a customer churn prediction model is a real telco dataset that includes details about Orange's fiber customers, such as their demographics, usage, and a target about whether they left Orange.

A closer look shows that the dataset contains 21 columns (features) and 94473 rows.

The goal is to use the data to explore the customer churn or not. To make that we need Build an ML model that predicts customer churn or not.

In this case the target takes a value of (0, 1).

0 ➔ not churn customers.

1 ➔ churn customers.

Here's an overview of the steps we'll take in this article.

Let’s rock it!

# 2. Importing and understanding data

## 2.1 import library’s

**When use python to building you will use some libraries in this case this is the library used.**

```
#Importing required libraries
import pandas as pd
import numpy as np
import pickle
from sklearn.preprocessing import *
from sklearn.ensemble import *
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import *
from sklearn import preprocessing
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2,mutual_info_classif
from sklearn.metrics import accuracy_score,precision_score
import warnings
warnings.filterwarnings("ignore")
```

Figure (library used)

## 2.2 loading dataset

After the import libraries now take a look to dataset.

Now, we'll load the dataset from the Xlsx file into our Python file using Panda's library.

Code:

```
In [18]: fiber=pd.read_excel('FTTH-DataSet.xlsx')
         fiber.head()
```

Figure (read xlsx)

Output:

Out[2]:

| | ID | GOVERNORATE | Customer with orange_MONTHS | CUSTOMER_AGE_MONTHS | CUSTOMER_GENDER | COMMITMENT | COMMITMENT_FG | OF_SPEED | OF_PREV_SPEED | M |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | West Amman | 48.741935 | 567.677419 | M | 24 | 1 | 200 | 100.0 | |
| 1 | 2 | West Amman | 44.838710 | 740.580645 | M | 24 | 0 | 100 | 100.0 | |
| 2 | 3 | West Amman | 44.612903 | 531.096774 | M | 24 | 1 | 200 | 100.0 | |
| 3 | 4 | Balqa | 43.741935 | 645.612903 | M | 24 | 0 | 200 | 100.0 | |
| 4 | 5 | West Amman | 41.548387 | 311.774194 | M | 12 | 1 | 400 | 100.0 | |

Figure (output).

## 2.3 explanation some columns in dataset

This part talking about what these columns mean.

- Customer with Orange Months: Number of months that customers are dealing with orange.
- COMMITMENT_FG: represent if the customer takes the offer with commitment (1) or not (0).
- OF_SPEED: The current internet fiber speed.
- OF_PREV_SPEED: The previous internet speed.
- MIGRATION_FLAG: The customer benefits from the company's offerings.
- LAST_LINK_STATUS: The last status of the internet while the customers were connected.
- GB_TOTAL_CONSUMPTION_Month1: The number of GB consumed by the customer in the first month.
- TARGET: Whether or not customers churned (0: no churn, 1: churn).

## 3. Data cleaning.

Data cleaning is a very important and useful step; it makes workflow easier and more efficient. During this process, we did some things:

- Dropping some columns / (Cell No.3)

  **NOTE: talking about this in features selection**

- Remove some rows that have illogical data. / (Cell No.5)

  Some features contain illogical data for example:

  Customers who disconnect more than 1000 times per month.

- Filling null values with zeros / (Cell No.4)
- Encoding (one hot encoding and manual encoding) / (Cell No. 8+9)

## 4. Detect imbalances in data

In calcification dataset should know if the data you dealing with is balanced or not, this function till us that data balanced or not, the function is:

Find_imbalanced (name, value1, value2)

Target column's name

Value 1: The value of the class (1 or 2).

Value2: The value of the class (1 or 2).

In our data, the ratio of 1 over 0 was 0.005, indicating an imbalance.

# 5. Spliting data and Handling the imbalanced using compute_class_weight().

## def split(data,col):

This function works to spliting the data to X and y and use train_test_split () that mean:

- Set X as all selected features and Y as the target by dropping it from the features.
- To split the data into training and testing, use train_test_split ().

Inputs :

Data : Dataset.

Col : Target column.

Outputs:

The function return X_train, X_test, Y_train, and Y_test, after undersampling.

Following that, we use compute_class_weight method to correct the imbalance.

```
#imbalanced fix method
weights = compute_class_weight('balanced', classes=y_train.unique(), y=y_train)
weights=dict(enumerate(weights))
```

- Class weight: balanced.
- Classes: y_train.unique().
- Y : Original class label .

# 6. Feature Selection

For feature selection, we used SelectKBest with chi2 to select the top 8 features with the lowest p-values.

After that according output the feature is:

- GOVERNORATE.
- Customer with Orange Months.
- COMMITMENT_FG.
- OF_SPEED.
- OF_PREV_SPEED.
- MIGRATION_FLAG.
- GB_TOTAL_CONSUMPTION_Month1.

# 7. Building and evaluating the model

At this step, we are ready to build the model and select the best algorithms that give a more efficient score.

Basically, we are dealing with a binary classification problem, and our target is (0,1), so we used the Random Forest clarifier algorithm.

## 7.1 function (modeling).

That function works to takes the algorithm you want to test on your model.

It works to some steps:

- Take algorithm.
- Fit (X_train, y_train).
- predict(X_test).

Input:

Model: the algorithms won't use it.

Output:

- classification report.
- accuracy for test.
- accuracy for train.

## 7.2 Testing part.

### 7.2.1 Default Hyperparameters.

In the algorithm testing part, we used (LogisticRegression and RandomForestClassifier) with default Hyperparameters and implemented them into the model, and we got these results.

```
LogisiticRegression
accuracy for test : 0.82
accuracy for train : 0.82

 RandomForestClassifier
 accuracy for test : 0.99
 accuracy for train : 1.0
```

Figure (default Hyperparameters result)

### 7.2.2 grid_search function.

To make the proses easier we create function for GridSearchCV this function take input:

- model = algorithm that use
- prameters= Hyperparameters
- Scoring= the score we looking to

Output:

best_params_= the best Hyperparameters to use it in the model

in this case the Hyperparameters is:

for RandomForestClassifier is:

('criterion':'gini,' max_depth: 6, max_features:'auto,' and n_estimators:50)

for LogisiticRegression is:

(solver='liblinear')

### 7.2.3 With hyperparameters.

After used the grid search method to get the best parameters to get more accuracy for both algorithms before choosing the final model, and here are the results.

```
LogisiticRegression
accuracy for test : 0.82
accuracy for train : 0.82

RandomForestClassifier
accuracy for test : 0.83
accuracy for train : 0.83
```

Figure (with Hyperparameters result)

In the end, we choose the Random Forest algorithm with these parameters.

('criterion': 'gini,' max_depth: 6, max_features: 'auto,' and n_estimators: 100) in the final model

## 8. Save the model.

After creating two models and doing enhancement on it, it has been observed that RandomForest has accuracy (0.80) higher than LogisiticRegression (0.79) so it will be taken. Then, we used Pickle to save our model in order to make the final model more user-friendly.

# 9. creating user interface

## 9.1 Description

In this part we show the steps in user notebook, it will take any input for information that related to the house that user want and give him the predation for customers, the file that has the best model has been loaded into user notebook.

this function “predict “has been created:

- load the best model using pickle
- the function to make the GUI for user

that function has two parts:

- GUI input
- Feature value

The output is:

prediction the target from the GUI input

## 9.2 how is works

1) input_data=np. array(input)= take a value from GUI interface
2) input_data=pd.DataFrame(input_data) = convert to data frame
3) input_data=input_data.transpose() convert rows to columns and vice versa
4) df1=pd.get_dummies (pd. DataFrame ({'GOVERNORATE’: [governorate_name_entry.get ()]})) = chose the governorate name
5) re_data=df1.reindex(columns=pd.get_dummies(df, columns=["GOVERNORATE"]).columns, fill_value=0)= make the dataframe for one hot encoding in column (governorate)
6) re_data=re_data.iloc[:,7:]= selecting the name of governorate form all data
7) test_user=pd.concat([input_data,re_data],axis=1)= margin the data from one hot encoding and other Feature in one data frame
8) prediction = model.predict(test_user)= to show the prediction output

How to deal with User Interface:

1. open Orange interface web.

2. Run the notebook.

3. Enter user data and click ( Enter data) to show the predict result.

# 10. Results

1. For this case we detect imbalance issue in the data so, the ratio was (0.005) to fix that we used under sampling method to make ratio reasonable.
2. Feature selection give us the most important features that affect on our target and we ignore the rest in order to get the most accurate result.
3. For building the best model for this data we followed many steps
    a. Functionality the processes to make testing easier
    b. Testing two algorithms with hyperparameters for this algorithm
    c. We used grid search to get best hyperparameters for these algorithm
    d. Implement hyperparameters that we got and compare the results for the algorithm.
    e. We chose Random forest classifier according the result we got.
4. To make this model useable we followed two steps:
    i. Using pickle library to save the final model we chose
    ii. The pickle file we saved get us the opportunity to use it in GUI we create
    iii. The GUI we have in few word we can say:
        1. We can enter the customer data we have
        2. That data we can use it to predict the probability if the customer will churn or not.

# 11. Conclusion

Customer churn prediction is crucial to the long-term financial stability of a company. In this project, we successfully created a machine learning model that's able to predict fiber customer churn with an accuracy of 83%.

There are many factors we can’t control to make these factor better such as customer behavior and customer financial, these factor has strong relationship if the customer will churn or not.