Group FTR

Week 9: Deliverables (Pls Scroll down)

Team Details

Name	Email	Country	College/Company	Specialization
Fabian Umeh	Fabianumeh335@gmail.com	UK	Teesside University	Data Science
Rukevwe Ovuowo	rukevwe10@gmail.com	Nigeria	GBG Data science Academy	Data Science
Olutayo Oladeinbo	oladeinboolutayo@yahoo.com	UK	Teesside University	Data Science

Problem statement (Week 7)

Churn rate is a marketing metric that describes the number of customers who leave a business over a specific time. Every user is assigned a prediction value that estimates their state of churn at any given time.

Business Understanding

Browsing behaviour Historical purchase data among other information It factors in our unique and proprietary predictions of how long a user will remain a customer. This score is updated every day for all users who have a minimum of one conversion. The values assigned are between 1 and 5.

Project lifecycle

Two weeks—deadline (1/09/2022)

Data intake report

Name: Customer Churn score prediction
Report date: 18/08/2022
Internship Batch: LISUM11: 30
Version:<1.0>
Data intake by: Fabian Umeh, Rukevwe Ovuowo, and Olutayo Oladeinbo
Data intake reviewer: Group members
Data storage location: [Github]
Tabular data details:
Total number of observations: 36992
Total number of files: 1
Total number of features: 25
Base format of the file: .csv
Size of the data: 8.3 MB

PROBLEMS IN DATA (Week 8)

1.1 Missing values:

Some values in columns such as [region_category, preferred_offer_types, points_in_wallet], appears to be missing.

1.2 Approach:

- The region category was encoded as follows:

City – 3

Town – 2

Village – 1

And as such, missing values were filled with the non-extreme value (2) for town.

- The points_in_wallet column had a relatively small percentage of missing values, and these values were filled using the average of the collected samples.

- The Preferred offer type column was encoded as follows:

Offer – 1

Without offer – 0

And as such the few missing values were replaced with no offers.

```
[ ] new= new.fillna({'region_category' : 2,'points_in_wallet' : new['points_in_wallet'].mean(),
                     'preferred_offer_types' : 0})
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36992 entries, 0 to 36991
Data columns (total 25 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   customer_id                   36992 non-null  object 
 1   Name                          36992 non-null  object 
 2   age                           36992 non-null  int64  
 3   gender                        36992 non-null  object 
 4   security_no                   36992 non-null  object 
 5   region_category               31564 non-null  object 
 6   membership_category           36992 non-null  object 
 7   joining_date                  36992 non-null  object 
 8   joined_through_referral       36992 non-null  object 
 9   referral_id                   36992 non-null  object 
 10  preferred_offer_types         36704 non-null  object 
 11  medium_of_operation           36992 non-null  object 
 12  internet_option               36992 non-null  object 
 13  last_visit_time               36992 non-null  object 
 14  days_since_last_login         36992 non-null  int64  
 15  avg_time_spent                36992 non-null  float64
 16  avg_transaction_value         36992 non-null  float64
 17  avg_frequency_login_days      36992 non-null  object 
 18  points_in_wallet              33549 non-null  float64
 19  used_special_discount         36992 non-null  object 
 20  offer_application_preference  36992 non-null  object 
 21  past_complaint                36992 non-null  object 
 22  complaint_status              36992 non-null  object 
 23  feedback                      36992 non-null  object 
 24  churn_risk_score              36992 non-null  int64  
dtypes: float64(3), int64(3), object(19)
memory usage: 7.1+ MB
```

2.1 String Error:

The data type in the average frequency login days column was supposed to be 'int' but rather contained some string value 'Error', which was further removed.

2.2 Approach:

The average frequency login days column keeps record of the average days the company site is visited by a customer. So ideally 'Error', most likely represent 0 login days was replaced as such.

Inspection of avg_frequency_login_days for posible anomaly:

From findings, the column contains 'int' and 'str' values, and as a result was not considered by the describe function above.

```
[ ] #Detection of str value from avg_frequency_login_days column
    print(set([data for data in data['avg_frequency_login_days'] for a in data if a \
        not in ['0','1','2','3','4','5','6','7','8','9','.','-']]))

    {'Error'}
```

```
[ ] #replacing error(str) values with 0
    data = data.replace({'avg_frequency_login_days': {'Error': 0}})
    #converting to integer
    data['avg_frequency_login_days'] = data['avg_frequency_login_days'].astype('float64')
```

3.1 Negative values:

Some columns contained negative values, which appeared to be anomalous and were further excluded from the analysis.

3.2 Approach:

The negative (anomalous) values were few and were excluded from the analysis.

Check and removal of negative values (anomalous values):

```
#code to check for the negative values in the Dataframe we noticed from the describe function
anom = data[['avg_time_spent','days_since_last_login','points_in_wallet','churn_risk_score','avg_frequency_login_days']].min(axis=0)
anom[anom < 0]

avg_time_spent             -2814.109110
days_since_last_login       -999.000000
points_in_wallet            -760.661236
churn_risk_score              -1.000000
avg_frequency_login_days     -43.652702
dtype: float64
```

```
[ ] #code to remove negative values in the avg_time_spent column
    data = data.drop(data[data['avg_time_spent'] < 0].index).copy()

    #code to remove negative values in the avg_time_spent column
    data = data.drop(data[data['days_since_last_login'] < 0].index).copy()

    #code to remove negative values in the avg_time_spent column
    data = data.drop(data[data['points_in_wallet'] < 0].index).copy()

    #code to remove negative values in the avg_time_spent column
    data = data.drop(data[data['churn_risk_score'] < 0].index).copy()

    #code to remove negative values in the avg_time_spent column
    new_data = data.drop(data[data['avg_frequency_login_days'] < 0].index).copy()
```

4.1 Duplicate check:

Duplicate check:

```
[ ] print('counting duplicates')
    len(new_data) - len(new_data.drop_duplicates())

    counting duplicates
    0
```

Data Transformations (Week 9)

1. Removing Unused columns

Some features appeared to be irrelevant in the prediction of customers churn score rate, and as such were excluded from the study.

```
#dropping unused columns
new_data = new_data.drop(['security_no', 'Name', 'customer_id','referral_id'], axis =1).copy()
```

2. Feature Engineering

The data in its raw form can only provide limited information. Some transformation techniques were performed on the data, this is to convert the data into a machine-readable format. Some of the transformation includes: date formatting, binary encoding and categorical encoding.

- Date Formatting

```
#converting to datatime format
new_data['joining_date'] = new_data['joining_date'].apply(pd.to_datetime)
```

- Getting the number of days spent by each customer

```
# Getting the number of days customer has spent with the company

#code to convert from time stamp in days to integer
new_data['Days_w_com'] = (new_data['joining_date'] - new_data['joining_date'].min()

#convert to string
new_data['Days_w_com'] = new_data['Days_w_com'].astype('str')

#split and pick only integer values
new_data['Days_w_com'] = new_data['Days_w_com'].str.split(" ", n=1, expand=True)[0]

#replace the latest value with 0
new_data['Days_w_com'] = new_data['Days_w_com'].replace('NaT', 0)

#convert to integer
new_data['Days_w_com'] = new_data['Days_w_com'].astype('int')
```

- Getting the number of hours spent in the last visit by each customer

```
#gettint the hours from last time visit
last_timv = new_data['last_visit_time'].str.split(r":", expand=True)

#fill Nan values with zero before computing hour
new_data['hour'] = last_timv[0].fillna(0).astype('int')
new_data['min'] = last_timv[1].fillna(0).astype('int')
new_data['sec'] = last_timv[2].fillna(0).astype('int')
new_data['hour'] = new_data['hour'] + (new_data['min']/60) + new_data['sec']/3600
```

- Low-level categorical feature format

Some categorical features has some classes either few or unknown. To handle this, the said classes were split equally between the two major class as shown below.

```
#code to get the dataframe with unknown gender
unknown = new[new['gender']== 'Unknown']

#code to divide the unkown into two equal part
index1= new[new['gender']== 'Unknown'][:len(unknown) // 2].index
index2= new[new['gender']== 'Unknown'][len(unknown) // 2 :].index
#code for allocating  Each half
new.loc[index1,'gender'] =  'M'

new.loc[index2,'gender'] =  'F'

#code to get the dataframe with unknown referral status
Un_ref = new[new['joined_through_referral']== '?']

#code to divide the unkown into two equal part
index1= new[new['joined_through_referral']== '?'][:len(Un_ref) // 2].index
index2= new[new['joined_through_referral']== '?'][len(Un_ref) // 2 :].index

#code for allocating  Each half to Yes and No
new.loc[index1,'joined_through_referral'] =  'Yes'
new.loc[index2,'joined_through_referral'] =  'No'
```

- Generalizing categories to reduce complexity

Classes that represent the same phenomena were further classified as one, for instance the feedback columns with categories such as ‘Poor Product Quality’, ‘Poor Website’, ‘Too many ads’, ‘Poor Customer Service’ all represent bad reviews.

```
#replacing values in columns for better understanding and reduced dimension
new = new.replace({'complaint_status': {'Solved in Follow-up': 'Solved',
                                        'No Information Available': 'Not Applicable'}})

new = new.replace({'feedback': {'Poor Product Quality': 'Bad',
                                'Poor Website': 'Bad',
                                'Too many ads': 'Bad',
                                'Poor Customer Service': 'Bad',
                                'Reasonable Price': 'Good',
                                'User Friendly Website': 'Good',
                                'Products always in Stock': 'Good',
                                'Quality Customer Care': 'Good',
                                'No reason specified': 'Good'
                                }})

new = new.replace({'medium_of_operation': {'?': 'Both'}})

new = new.replace({'preferred_offer_types': {'Gift Vouchers/Coupons': 'Offer',
                                             'Credit/Debit Card Offers': 'Offer'}})
```

- Binary Encoding

```
#Encoding Binary columns
new['gender'] = new['gender'].apply(lambda x: 1 if x =='M' else (0 if x =='F' else None))
new['joined_through_referral'] = new['joined_through_referral'].apply(lambda x: 1 if x =='Yes' else (0 if x =='No' else None))
new['feedback'] = new['feedback'].apply(lambda x: 1 if x =='Good' else (0 if x =='Bad' else None))
new['used_special_discount'] = new['used_special_discount'].apply(lambda x: 1 if x =='Yes' else (0 if x =='No' else None))
new['past_complaint'] = new['past_complaint'].apply(lambda x: 1 if x =='Yes' else (0 if x =='No' else None))
new['offer_application_preference'] = new['offer_application_preference'].apply(lambda x: 1 if x =='Yes' else (0 if x =='No' else None))
new['preferred_offer_types'] = new['preferred_offer_types'].apply(lambda x: 1 if x =='Offer' else (0 if x =='Without Offers' else None))
```

- Categorical encoding

```
new = new.replace({'region_category': {'City': 3, 'Town': 2, 'Village':1}})
```

```
new = new.replace({'membership_category': {'No Membership': 0, 'Basic Membership': 1,
                                           'Premium Membership':2,'Silver Membership':3,
                                           'Silver Membership':4,'Gold Membership':5,
                                           'Platinum Membership':6}})
```

3. Finally, Handling of missing values

```
new= new.fillna({'region_category' : 2,'points_in_wallet' : new['points_in_wallet'].mean(),
                 'preferred_offer_types' : 0})
```

```
new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 31618 entries, 0 to 36991
Data columns (total 21 columns):
 #   Column                        Non-Null Count  Dtype   
---  ------                        --------------  -----   
 0   age                           31618 non-null  int64   
 1   gender                        31618 non-null  int64   
 2   region_category               31618 non-null  float64 
 3   membership_category           31618 non-null  int64   
 4   joined_through_referral       31618 non-null  int64   
 5   preferred_offer_types         31618 non-null  float64 
 6   medium_of_operation           31618 non-null  object  
 7   internet_option               31618 non-null  object  
 8   days_since_last_login         31618 non-null  int64   
 9   avg_time_spent                31618 non-null  float64 
 10  avg_transaction_value         31618 non-null  float64 
 11  avg_frequency_login_days      31618 non-null  float64 
 12  points_in_wallet              31618 non-null  float64 
 13  used_special_discount         31618 non-null  int64   
 14  offer_application_preference  31618 non-null  int64   
 15  past_complaint                31618 non-null  int64   
 16  complaint_status              31618 non-null  object  
 17  feedback                      31618 non-null  int64   
 18  churn_risk_score              31618 non-null  category
 19  Days_w_com                    31618 non-null  int64   
 20  hour                          31618 non-null  float64 
dtypes: category(1), float64(7), int64(10), object(3)
memory usage: 6.1+ MB
```