data_analysis

April 29, 2022

1 Churn Project

1.1 Setup Environment

To install all the packages and their dependencies we ran the following commands via bash terminal

[2]:

```
#$ export ML_PATH="$HOME/ml_project" && mkdir -p $ML_PATH
#$ python3 -m pip install --user -U pip
#$ python3 -m pip install --user -U virtualenv
#$ cd $ML_PATH
#$ virtualenv ml_env
#$ source ml_env/bin/activate
#$ python3 -m pip install -U jupyter matplotlib seaborn numpy pandas scipy scikit-learn bioinfokit statsmodels imblearn
#$ python3 -m ipykernel install --user --name=python3
#$ jupyter notebook


#$ ctrl + c
#$ deactivate
```

1.2 Download and Load the Data

We download the data directly from the DropBox link and load them in the Jupyter workspace as Pandas Dataframe. We than call the .head() method to check the result.

[ ]:

```python
import os
import urllib
import pandas as pd

DOWNLOAD_URL = 'https://www.dropbox.com/s/7nwimmta836si5f/churn.csv?dl=1'
CHURN_PATH = os.path.join("dataset", "churn")

# Download data directly from Dropbox
def fetch_data(download_url=DOWNLOAD_URL, path=CHURN_PATH):
    os.makedirs(path, exist_ok=True)
    csv_path = os.path.join(path, "churn.csv")
    urllib.request.urlretrieve(download_url, csv_path)
```

```
# Load data
def load_data(path=CHURN_PATH):
    csv_path = os.path.join(path, "churn.csv")
    return pd.read_csv(csv_path)

fetch_data()
churn = load_data()

# Check result
churn.head(3)
```

[ ]:

```
# Pop a couple of vars and save to var to use later
CLIENTNUM = churn.pop('CLIENTNUM')
Unamed = churn.pop('Unnamed: 0')

churn.head(3)
```

[ ]:

```
churn1=churn
```

1.3 Data Exploration

We ran a few methods just to get a quick grasp of the data. Spcifically we wanted ot inspect:

- The number of columns and rows
- The variables and their data types
- The number of non-null values
- The categories for each categorical feature
- The descriptive statistics for the numerical variables

[ ]:

```
churn.shape
```

[ ]:

```
# Quick vars description
churn.info()
```

[ ]:

```
# Value counts on categorical vars
not_num = ['Attrition_Flag', 'Gender', 'Education_Level', 'Marital_Status',␣
 ↪'Income_Category', 'Card_Category']

for var in not_num:
    print(churn[var].value_counts())
    print('\n')
```

[ ]:

```
# Basic descriptive stats
churn.describe()
```

1.4 Plots

Below we plot:

- the histograms of all the numerical variables in the dataset and we replot those that show artifacts or outliers to further inspect (Customer_Age, Months_on_Book)
- the target variable distribution via barplot
- the barplot of all the categorical variables
- the kde curves of Existing and Attrited customers for each variable

1.4.1 Histograms

[ ]:

```
%matplotlib inline
import matplotlib.pyplot as plt

# Lets plot a hist for each numerical attribute
churn.hist(bins=50, figsize=(15,20))
plt.show()
```

1.4.2 Futher analysis from hists insights

[ ]:

```
# COSTUMER AGE
# find and fix missing values (discovered if capped/preprocessed)

count = churn['Customer_Age'].value_counts()
min_age = churn['Customer_Age'].min()
max_age = churn['Customer_Age'].max()

# Print missing values if any between min and max
for i in range(min_age,max_age):
    if i not in count:
        print(i)

# Lets replot a better hist
plt.hist(x = churn.Customer_Age, bins = 100, color = 'green')
plt.title('Costumers Age')
plt.xlabel('Ages')
plt.ylabel('Count')
plt.show()

# It probably was a graphical artifact
```

[ ]:

```
# OUTLIERS
# Months on book outlier
plt.hist(x = churn.Months_on_book, bins = 100, color = 'green')
plt.title('Months on book')
plt.xlabel('N of months')
plt.ylabel('Count')
plt.show()

count = churn['Months_on_book'].value_counts()
count
# 36 months on book - may be an outlier but since it is exactly a 3 year period it might just be due to 3 years promo or plan

# CROSS TABLE con attrited - quanti ce ne sono che poi se ne sono andati
cross = pd.crosstab(churn['Months_on_book'],churn['Attrition_Flag'])
#cross # 430 costumers left after 36 months
```

1.4.3 Plot of Target Distribution

[ ]:

```
import seaborn as sns

def bar_plot(df,column):
    ax = sns.countplot(y=column, data=df)
    plt.title('{0} Percentage'.format(column))
    plt.xlabel('Count')
    total = len(df[column])
    for p in ax.patches:
        percentage = '{:.1f}%'.format(100 * p.get_width()/total)
        x = p.get_x() + p.get_width() + 0.02
        y = p.get_y() + p.get_height()/2
        ax.annotate(percentage, (x, y))
    plt.show()

bar_plot(churn, "Attrition_Flag")
```

1.4.4 Categorical variables plots

[ ]:

```
for var in not_num:
    bar_plot(churn, var)
```

[ ]:

```
# INCOME CATEGORY
# These categories are not in ORDINAL ORDER

import seaborn as sns

# Hist
plt.hist(x = churn.Income_Category, bins = 20, color = 'blue')
plt.title('Income Category')
plt.xlabel('Category')
plt.xticks(rotation=30)
plt.ylabel('Count')
sns.despine(top=True, right=True, left=False, bottom=False)
plt.show()
```

1.4.5 Kde of numerical vars

[ ]:

```
def kdeplot(feature, hist, kde):
    plt.figure(figsize=(10, 5))
    plt.title("Plot for {}".format(feature))
    ax0 = sns.distplot(churn[churn['Attrition_Flag'] == 'Existing␣
 ↪Customer'][feature], hist=hist, kde=kde,
                  color = 'darkblue',  label= 'Existing Customer',
                  hist_kws={'edgecolor':'black'},
                  kde_kws={'linewidth': 3})
    ax1 = sns.distplot(churn[churn['Attrition_Flag'] == 'Attrited␣
 ↪Customer'][feature], hist=hist, kde=kde,
                  color = 'orange',  label= 'Attrited Customer',
                  hist_kws={'edgecolor':'black'},
                  kde_kws={'linewidth': 3})
    #plt.savefig('kde.png')
    plt.legend()

attributes = ["Customer_Age", "Dependent_count", "Months_on_book",␣
 ↪"Total_Relationship_Count",
              "Months_Inactive_12_mon", "Contacts_Count_12_mon",␣
 ↪"Credit_Limit", "Total_Trans_Amt",
             "Total_Trans_Ct", "Avg_Utilization_Ratio"]

for attr in attributes:
    kdeplot(attr, hist = False, kde = True)
```

1.5 Cross Tables

We will compute crosstables mainly between the target variable, Attrition_Flag, and the other categorical variables. We also selected a few interestring continuos variables and cut them them in categories to make the crosstables interpretable.

[ ]:

```
# Lets create a copy of the dataset
churn0 = churn.copy()

# Cross table function
def cross_prop(var):
    cross = pd.crosstab(churn0[var],churn0['Attrition_Flag'])
    prop = cross['Attrited Customer']/cross['Existing Customer'] # Check for␣
 ↪prevalence of categories in attrited
    return cross, prop
```

```
# Plot cross prop
def plot_cross_prop(cross):
    return cross.div(cross.sum(1).astype(float), axis=0).plot(kind="bar",␣
 ↪stacked=True, figsize=(6, 6))
```

1.5.1 Categorical variables

[ ]:

```
# Income_Category - Attrition_Flag
cross, prop = cross_prop('Income_Category')
print(cross)
print('\n')
print(prop)
```

[ ]:

```
# Card_Category - Attrition_Flag
cross, prop = cross_prop('Card_Category')
print(cross)
print('\n')
print(prop)
print("\n\033[1mThere is a slight prevalence of Platinum and Gold users in the␣
 ↪attrited costumers\033[0m")

# Plot
plot_cross_prop(cross)
```

[ ]:

```
# Gender - Attrition_Flag
cross, prop = cross_prop('Gender')
print(prop)
print("\n\033[1mThere is a negligible prevalence of Females among the attrited␣
 ↪costumers\033[0m")
```

[ ]:

```
# Educational Level - Attrition_Flag
cross, prop = cross_prop('Education_Level')
print(prop)
print("\n\033[1mAttrition seems slightly higher among Doctorate\033[0m")

# Plot
plot_cross_prop(cross)
```

[ ]:

```
# Marital_Status - Attrition_Flag
cross, prop = cross_prop('Marital_Status')
print(prop)
```

1.5.2 Continuos variables

[ ]:

```
import numpy as np

# Quantile-based discretization
def quantcut(var):
    churn0[var] =  pd.qcut(
        churn0[var],  # Column to bin
        6)           # Number of quantiles
    return churn0[var]
```

[ ]:

```
churn0.head(3)
```

[ ]:

```
# Age
churn0['Customer_Age'] = quantcut('Customer_Age')
cross, prop = cross_prop('Customer_Age')

print(prop)
```

[ ]:

```
# Credit Limit
churn0['Credit_Limit'] = quantcut('Credit_Limit')
cross, prop = cross_prop('Credit_Limit')
print(cross)
print('\n')
print(prop)
```

[ ]:

```
# Total_Trans_Amt
churn0['Total_Trans_Amt'] = quantcut('Total_Trans_Amt')
cross, prop = cross_prop('Total_Trans_Amt')

print(prop)
```

1.6 Train-Test Split

Here we train-test split just to be able to run the correlation analyisis on the train set alone.

1.6.1 Plain

[ ]:

```
# Classic sklearn split
from sklearn.model_selection import train_test_split

train_set, test_set = train_test_split(churn, test_size=0.2, random_state=42)
print("\033[1mTrain:\033[0m", len(train_set), "\t\033[1mTest:\033[0m",␣
 ↪len(test_set))
```

1.6.2 Stratified split based on Attrition Flag

[ ]:

```
# Compute STRATIFIED SAMPLING on ATTRITION FLAG
from sklearn.model_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n_splits=1, test_size=.2, random_state=42)

for train_index, test_index in split.split(churn, churn["Attrition_Flag"]):
    strat_train_set = churn.loc[train_index]
    strat_test_set = churn.loc[test_index]

strat_train_set.head(5)
```

1.7 ## Correlation analysis

1.7.1 Pearson's R

We compute here the correlation matrix (r pearson coefficients) for all the varibles in the dataset. We had to factorize the categorical variables to be able to run this analysis on them as well. We ran this analysis on both the full dataset and on the sole train set. Moreover we recomputed the correlation matrix after creating a new featue called Amount per transaction (Amt_per_Trans) which consists of the ratio between Trans_Amt and Trans_Count. Finally we recomputed the matrix after deleting all the rows presenting "Unkown" values.

[ ]:

```
import seaborn as sns

#churn = train_set.copy()

# Compute corr matrix
churn_corr = churn.apply(lambda x: pd.factorize(x)[0])   # factorize to include␣
 ↪categorical
corr_matrix = churn_corr.corr()

plt.figure(figsize=(12,8))
sns.heatmap(corr_matrix, cmap="YlGnBu", annot=True)  # alternative:␣
 ↪cmap='Blues_r'
```

[ ]:

```
# Check correlation scores for Attrition
corr_matrix["Attrition_Flag"].sort_values(ascending=False)
```

1.7.2 Create new attribute and recompute correlation

[ ]:

```
# Total trans amt and count
churn["Amt_Per_Trans"] = churn["Total_Trans_Amt"] / churn["Total_Trans_Ct"]
#churn["Card*Credit"] = churn["Card_ordinal"] * churn["Credit_Limit"]

# Compute corr matrix
```

```
churn_corr = churn.apply(lambda x: pd.factorize(x)[0])   # factorize to include␣
 ↪categorical
corr_matrix = churn_corr.corr()

plt.figure(figsize=(12,8))
sns.heatmap(corr_matrix, cmap="YlGnBu", annot=True)  # alternative:␣
 ↪cmap='Blues_r'
```

[ ]:

```
corr_matrix["Attrition_Flag"].sort_values(ascending=False)
```

[ ]:

```
# Replace Unkown with NAN
import numpy as np

unkown_vars = ['Attrition_Flag','Education_Level', 'Marital_Status',␣
 ↪'Income_Category']

def replace_unkown(dataset):

    for var in unkown_vars:
        if var in dataset.columns:
            dataset[var] = dataset[var].replace("Unknown", np.NaN)
    return dataset.dropna()

data_full = replace_unkown(churn).reset_index(drop=True)

# Compute corr matrix
churn_corr = data_full.apply(lambda x: pd.factorize(x)[0])   # factorize to␣
 ↪include categorical
corr_matrix = churn_corr.corr()

plt.figure(figsize=(12,8))
sns.heatmap(corr_matrix, cmap="YlGnBu", annot=True)
```

1.8 Chi-squared test

We used Pearson's Chi-Squared Test and Likelihood Chi-Squared Test to test for the dependence between "Attrition_Flag" and the others categorical variables in the dataset. We used a function in the python package "bioinfokit.analys" to perform these tests. The output of the function provides Degrees of Freedom and p-values. In order to perform the Chi_Squared Test of Independence we used cross tables between our target variable "Attrition_Flag" and each of that categorical variables.

[ ]:

```
from bioinfokit.analys import stat
categorial_var=["Gender","Education_Level","Marital_Status","Card_Category","Income_Category"]
for cat in categorial_var:
    print(cat)
    cross = pd.crosstab(churn1['Attrition_Flag'],churn1[cat])
    #print(cross)
    res = stat()
    res.chisq(df=cross)
    print(res.summary)
```

[ ]:

```
unkown_vars = ['Education_Level', 'Marital_Status', 'Income_Category']

def replace_unkown(dataset, var):
    if var in dataset.columns:
        dataset[var] = dataset[var].replace("Unknown", np.NaN)
    return dataset.dropna()

for var in unkown_vars:
    print(var)
    data_full = replace_unkown(churn1, var).reset_index(drop=True)
    cross = pd.crosstab(data_full['Attrition_Flag'],data_full[var])
    #print(cross)
    res = stat()
    res.chisq(df=cross)
    print(res.summary)
```

1.9 ANOVA test

We performed the Anova test to analyzes the relationship between the categorical target variable and the individual quantitative variables of the dataset.We used a function from the python package “statsmodel” that runs a F test. As a way to visualize and better understand the relationship between “Attrition_Flag” and quantitative variables, we also plotted boxplots.

[ ]:

```
import statsmodels.api as sm
from statsmodels.formula.api import ols

quantitative_var=["Customer_Age","Dependent_count","Months_on_book","Total_Relationship_Count'
for var in quantitative_var:
    reg=var+'~ Attrition_Flag'
    mod = ols(reg,data=churn1).fit()
    aov_table = sm.stats.anova_lm(mod, typ=2)
    print(var)
    print(aov_table)
    print("\n")
```

1.9.1 Box Plots

[ ]:

```
for var in quantitative_var:
    plot=churn1.boxplot(var,by="Attrition_Flag")
    print(plot)
```