CS3802--Machine Learning Algorithms Lab

Adithya V | BTech CSE (IoT) - A | 21011102009

Exercise 3

Use the dataset, perform necessary pre-processing and build a logistic regression model. divide the data into 70:30 ratio and print the performance metrics.

Importing the necessary libraries

In [ ]:
```
import pandas as pd
from statsmodels.stats.outliers_influence import variance_inflation_factor as VIF
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error, r2_score
import math
from sklearn.preprocessing import RobustScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Reading the dataset

In [ ]:
```
data = pd.read_csv('telecom_customer_churn.csv')
data.head()
```

Out[ ]:

	Customer ID	Gender	Age	Married	Number of Dependents	City	Zip Code	Latitude	Longitude	Numl Referı
0	0002-ORFBO	Female	37	Yes	0	Frazier Park	93225	34.827662	-118.999073	
1	0003-MKNFE	Male	46	No	0	Glendale	91206	34.162515	-118.203869	
2	0004-TLHLJ	Male	50	No	0	Costa Mesa	92627	33.645672	-117.922613	
3	0011-IGKFF	Male	78	Yes	0	Martinez	94553	38.014457	-122.115432	
4	0013-EXCHZ	Female	75	Yes	0	Camarillo	93010	34.227846	-119.079903	

5 rows × 38 columns

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6589 entries, 0 to 6588
Data columns (total 38 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   Customer ID                        6589 non-null   object
 1   Gender                             6589 non-null   object
 2   Age                                6589 non-null   int64
 3   Married                            6589 non-null   object
 4   Number of Dependents               6589 non-null   int64
 5   City                               6589 non-null   object
 6   Zip Code                           6589 non-null   int64
 7   Latitude                           6589 non-null   float64
 8   Longitude                          6589 non-null   float64
 9   Number of Referrals                6589 non-null   int64
 10  Tenure in Months                   6589 non-null   int64
 11  Offer                              6589 non-null   object
 12  Phone Service                      6589 non-null   object
 13  Avg Monthly Long Distance Charges  5945 non-null   float64
 14  Multiple Lines                     5945 non-null   object
 15  Internet Service                   6589 non-null   object
 16  Internet Type                      5245 non-null   object
 17  Avg Monthly GB Download            5245 non-null   float64
 18  Online Security                    5245 non-null   object
 19  Online Backup                      5245 non-null   object
 20  Device Protection Plan             5245 non-null   object
 21  Premium Tech Support               5245 non-null   object
 22  Streaming TV                       5245 non-null   object
 23  Streaming Movies                   5245 non-null   object
 24  Streaming Music                    5245 non-null   object
 25  Unlimited Data                     5245 non-null   object
 26  Contract                           6589 non-null   object
 27  Paperless Billing                  6589 non-null   object
 28  Payment Method                     6589 non-null   object
 29  Monthly Charge                     6589 non-null   float64
 30  Total Charges                      6589 non-null   float64
 31  Total Refunds                      6589 non-null   float64
 32  Total Extra Data Charges           6589 non-null   int64
 33  Total Long Distance Charges        6589 non-null   float64
 34  Total Revenue                      6589 non-null   float64
 35  Customer Status                    6589 non-null   int64
 36  Churn Category                     1869 non-null   object
 37  Churn Reason                       1869 non-null   object
dtypes: float64(9), int64(7), object(22)
memory usage: 1.9+ MB
```

Pre-Processing

Remove outliers using IQR method

The `IQR_Removal` function takes a DataFrame (`df`) as input and performs the following steps:

1. **Initialization:**

- Retrieves column names from the DataFrame (`columns = df.columns`).

2. **Iterate Through Columns:**

   - For each column (`col`) in the DataFrame:
     - Skips the 'SalePrice' column.

3. **Check Column Type:**

   - Verifies if the column is not of type 'object' (i.e., numerical).

4. **Remove Outliers Using IQR:**

   - Calculates the first quartile (Q1), third quartile (Q3), and Interquartile Range (IQR) for the numerical column.
   - Filters rows to keep only those within the range of `(Q1 - 1.5 * IQR)` to `(Q3 + 1.5 * IQR)` .

5. **Return Updated DataFrame:**

   - Returns the DataFrame with outliers removed from numerical columns.

In [ ]:
```
def IQR_Removal(df):

    columns = df.columns
    for col in columns:
        if col == 'SalePrice':
            continue
        typeCol = str(df[col].dtype)
        if typeCol != 'object':
            Q1 = df[col].quantile(0.25)
            Q3 = df[col].quantile(0.75)
            iqr = Q3 - Q1
            df = df[(df[col] >= Q1 - 1.5*iqr) & (df[col] <= Q3 + 1.5*iqr)]
    return df
```

Remove columns with only one unique value and columns with null ratio >= 0.30

The `ThresholdandND_columnRemoval` function takes a DataFrame (`df`) as input and performs the following steps:

1. **Calculate Length and Columns:**

   - Retrieves the length of the DataFrame (`N = len(df)`) and column names (`columns = df.columns`).

2. **Iterate Through Columns:**

   - For each column (`col`) in the DataFrame:
     - Checks if the number of unique values in the column is equal to 1. If true, drops the column as it lacks diversity.

3. **Check Null Ratio:**

- Calculates the ratio of null values in the column (`notnull = df[col].isnull().sum()`) and checks if it is greater than or equal to 30%.
- If true, drops the column as it exceeds the specified null ratio threshold.

4. **Return Updated DataFrame:**

- Returns the DataFrame with columns removed based on the threshold and no-diversity criteria.

In [ ]:
```
def ThresholdandND_columnRemoval(df):

    N = len(df)
    columns = df.columns
    for col in columns:
        if (len(df[col].unique()) == 1):
            df = df.drop([col], axis=1)
            continue
        notnull = df[col].isnull().sum()
        ratio = notnull / N
        if (ratio >= 0.30):
            df = df.drop([col], axis=1)
    return df
```

Handle null values by either removing rows or filling with mean/median

The `Handling_NullValues` function takes a DataFrame (`df`) as input and performs the following steps:

1. **Iterate Through Columns:**

- For each column (`col`) in the DataFrame:
  - Checks the data type of the column (`typeCol = str(df[col].dtype)`).

2. **Handle Null Values for Object Type:**

- If the column type is 'object' (categorical):
  - Removes rows with null values for that column (`df = df[df[col].notna()]`).

3. **Handle Null Values for Numeric Type:**

- If the column type is numeric:
  - Calculates mean, median, and standard deviation of the column (`mean = df[col].mean()` , `median = df[col].median()` , `standard_deviation = df[col].std()`).

4. **Partial Median Change (PMC) Criteria:**

- Calculates Partial Median Change (PMC) using the formula `pmc = (3 * (mean - median)) / standard_deviation` .
- If PMC is greater than or equal to 0.4 or less than or equal to -0.4:
  - Fills null values with the median (`df[col] = df[col].fillna(median)`).

- Otherwise:
  - Fills null values with the mean (`df[col] = df[col].fillna(mean)`).

5. **Return Updated DataFrame:**

   - Returns the DataFrame with missing values handled based on data type and PMC criteria.

In [ ]:
```
def Handling_NullValues(df):

    columns = df.columns
    for col in columns:
        typeCol = str(df[col].dtype)
        if typeCol == 'object':
            df = df[df[col].notna()]
        else:
            mean = df[col].mean()
            median = df[col].median()
            standard_deviation = df[col].std()
            pmc = (3 * (mean - median)) / standard_deviation
            if pmc >= 0.4 or pmc <= -0.4:
                df[col] = df[col].fillna(median)
            else:
                df[col] = df[col].fillna(mean)
    return df
```

Perform one-hot encoding for categorical columns

The `OneHotEncoding_objects` function encodes categorical (object-type) columns using one-hot encoding:

1. **Iterate Through Columns:**

   - For each column (`col`) in the DataFrame:
     - Check if the column type is 'object'.

2. **One-Hot Encode Object Columns:**

   - If the column is 'object':
     - Use `pd.get_dummies` to create one-hot encoded columns.

3. **Rename and Join Encoded Columns:**

   - Rename the new columns by appending the original column name as a prefix.
   - Join the one-hot encoded columns to the original DataFrame.

4. **Drop Original Object Column:**

   - Drop the original object-type column.

5. **Return Updated DataFrame:**

   - Returns the DataFrame with one-hot encoded object-type columns.

In [ ]:
```
def OneHotEncoding_objects(df):

    columns = df.columns
    for col in columns:
        typeCol = str(df[col].dtype)
        if typeCol == 'object':
            enc = pd.get_dummies(df[col])
            encCol = enc.columns
            newColumns = {}
            for i in range(0, len(encCol)):
                newColumns[encCol[i]] = col + encCol[i]
            enc.rename(columns=newColumns, inplace=True)
            df = df.join(enc)
            df = df.drop([col], axis=1)
    return df
```

Result of Pre-Processing

In [ ]:
```
df = OneHotEncoding_objects(IQR_Removal(Handling_NullValues(ThresholdandND_columnRe
df.head()
```

Out[ ]:

	Age	Number of Dependents	Zip Code	Latitude	Longitude	Number of Referrals	Tenure in Months	Avg Monthly Long Distance Charges	Avg Monthly GB Download	Mon Ch
0	37	0	93225	34.827662	-118.999073	2	9	42.39	16.0	
2	50	0	92627	33.645672	-117.922613	0	4	33.65	30.0	
3	78	0	94553	38.014457	-122.115432	1	13	27.82	4.0	
6	67	0	93437	34.757477	-120.550507	1	71	9.96	14.0	1
8	68	0	93063	34.296813	-118.685703	0	7	10.53	21.0	

5 rows × 2894 columns

Robust Scaling

In [ ]:
```
scaler = RobustScaler()
cols = df.columns
data_scale = scaler.fit_transform(df.to_numpy())
data_scale = pd.DataFrame(data_scale, columns=cols)
data_scale
```

Out[ ]:

	Age	Number of Dependents	Zip Code	Latitude	Longitude	Number of Referrals	Tenure in Months	Avg Monthly Long Distance Charges	M Dou
0	-0.518519	0.0	-0.087323	-0.263945	0.127913	2.0	-0.289474	0.751416	
1	-0.037037	0.0	-0.273819	-0.548445	0.408869	0.0	-0.421053	0.384728	
2	1.000000	0.0	0.326836	0.503103	-0.685456	1.0	-0.184211	0.140130	
3	0.592593	0.0	-0.021207	-0.280838	-0.277011	1.0	1.342105	-0.609188	
4	0.629630	0.0	-0.137845	-0.391718	0.209703	0.0	-0.342105	-0.585274	
...	...	...	...	...	...	...	...	...	
2014	0.148148	0.0	0.760019	0.999556	-0.523307	0.0	0.210526	0.046570	
2015	0.962963	0.0	-0.433183	-0.760347	0.617682	1.0	0.868421	0.971680	
2016	-1.148148	0.0	-1.086231	-0.457428	0.347803	0.0	-0.342105	0.503881	
2017	0.074074	0.0	0.038360	0.212617	0.153368	0.0	-0.500000	0.738829	
2018	-0.407407	0.0	0.580695	0.435832	-0.382393	1.0	0.052632	-0.347388	

2019 rows × 2894 columns

Training the Model

In [ ]:
```
target = data_scale['Customer Status']
ivCol = list(data_scale.columns)
ivCol.remove('Customer Status')
independent_variables = data_scale[ivCol]
independent_variables
x_train, x_test, y_train, y_test = train_test_split(independent_variables, target,
logisticRegr = LogisticRegression()
logisticRegr.fit(x_train, y_train)
```

Out[ ]:
```
▾ LogisticRegression
LogisticRegression()
```

Preediction

In [ ]:
```
y_pred = logisticRegr.predict(x_test)
y_pred

accuracy = accuracy_score(y_test,y_pred)
accuracy
```

Out[ ]: 0.7788778877887789