<u>INF 428/528 Final Project</u>

Predictive Model for Service Calls Payment

Prediction Model

Yashoda Rishita Pothunuri
001570390

Introduction

In this project, we are provided information on clients that have a bill due in the next 5 days and regardless of whether they have called in a service payment. Our aim is to build a predictive model that detects clients who will probably make a service payment call (CALL_FLAG=1) within the next 5 days. This approach will be used to choose clients who can expect a preemptive e-mail urging individuals to make payments online.

Data Preparation

Preparing the data is the initial stage in creating a prediction model. To begin, we will import the data and look for values that are missing, outliers, and other data quality concerns. We employ the make_pipeline object and the SimpleImputer class to replace missing data with a defined value which includes the mean, median, or most common value or to transport the most recent known result forward or backward.

Dataset Variables

DATE_FOR: The date of observation

RTD_ST_CD: Code of the state

CustomerSegment: Customer segment number

Tenure: The number of months a client has been active.

Age: The customer's age.

MART_STATUS: Relationship status

GENDER: The gender of a person.

CHANNEL1_6M: The number of contacts with channel 1 during the previous six months.

CHANNEL2_6M: The number of contacts with channel 2 during the previous six months.

CHANNEL3_6M: The number of contacts with channel 3 during the previous six months.

CHANNEL4_3M: The number of contacts with channel 4 during the previous three months.

CHANNEL5_3M: The number of contacts with channel 5 during the previous three months.

METHOD1_3M: The number of interactions via method 1 in the previous three months.

PAYMENTS_3M: The number of payments made in the previous three months.

NOT_DI_3M: The total number of days since the customer interacted.

NOT_DI_6M: The total number of days since the customer interacted.

EVENT1_30_FLAG: Event 1 indicator variable over the previous 30 days.

EVENT2_90_SUM: Total number of events 2 in the previous 90 days.

LOGINS: The total number of logins in the previous six months.

POLICYPURCHASECHANNEL: The channel via which policies are purchased.

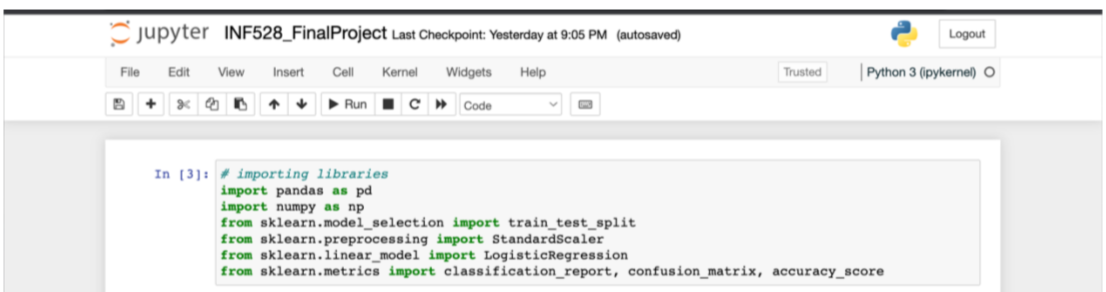
Call_Flag: Variable that indicates if a service payment call will be made in the next 5 days (goal variable).

Modeling Methods

A logistic regression model will be used to forecast the chance of each policyholder making a service payment call (CALL_FLAG=1). Logistic regression is a popular approach for binary classification issues because it produces transparent coefficients that help us comprehend the correlation between the input and output variables.

Workflow

The first line includes the pandas library, which is employed to manipulate and analyze data. The second line imports the numpy package, which is implemented in Python for scientific computation. The third line imports train_test_split from scikit-learn's model_selection module, which serves to split the dataset into training and testing sets. The fourth line imports StandardScaler from scikit-learn's preprocessing module, which functions for normalizing the feature variables. The fifth line imports LogisticRegression from scikit-learn's linear_model module, which serves to apply a logistic regression model to the data. The sixth line imports classification_report, confusion_matrix, and accuracy_score from scikit-learn's metrics module, which are used to evaluate the model's performance.




```python
# importing libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

This line reads data from a CSV file and writes it to a pandas dataframe named servicecallsdata.


```python
# loading the dataset
servicecallsdata = pd.read_csv('servicecallsdata.csv')
```

These two lines specify the columns in the dataframe servicecallsdata's feature variables (Features) and target variables (Target).

```
In [6]: # defining the features and target variables
Features = servicecallsdata[['Tenure', 'Age', 'CHANNEL1_6M', 'CHANNEL2_6M', 'CHANNEL3_6M', 'CHA
Target = servicecallsdata['Call_Flag']
```

This line splits the feature and target variables into training and testing sets using the train_test_split function, having a testing set size of 30% and a random state of 0 for repeatability.

```
In [7]: # spliting the data into training and testing sets
Features_train, Features_test, Target_train, Target_test = train_test_split(Features, Target, t
```

The following lines create a StandardScaler object and apply it to the training and testing sets in order to standardize the feature variables.

```
In [11]: # scaling the feature variables using standardization
ScalingFeature = StandardScaler()
Features_train = ScalingFeature.fit_transform(Features_train)
Features_test = ScalingFeature.transform(Features_test)
```

The lines below import the make_pipeline object and the SimpleImputer class in order to replace any missing values with the most frequently learnt value throughout the training phase.

```
In [12]: from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
```

```
In [13]: # creating a pipeline that imputes missing values and trains a logistic regression model
LogisticRegressionModel = make_pipeline(
    SimpleImputer(strategy='mean'),
    LogisticRegression(random_state=0)
)
```

The following lines create a LogisticRegression object and apply it to training data.

```
In [14]: # fiting the pipeline to the training data
         LogisticRegressionModel.fit(Features_train, Target_train)

Out[14]:  ▸     Pipeline
            ▸ SimpleImputer
          ▸ LogisticRegression
```

The trained model is used in this line to forecast the target variable for the testing set.

```
In [15]: # predicting the target variable for the test set
         Target_prediction = LogisticRegressionModel.predict(Features_test)
```

Results and Conclusion

These lines output the model's results on the testing set's confusion matrix, classification report, and accuracy score.

The model appears to be functioning satisfactorily in terms of predicting the negative category (0) with high precision and recall, as demonstrated by the confusion matrix and classification report. However, the model exhibits low accuracy and recall for the positive class (1), indicating that it is not effective at identifying clients who will probably make a service payment call. The accuracy score of 0.9626 is pretty good, however it might be deceptive in unbalanced datasets with a dominating negative class.

Overall, this code does a logistic regression evaluation on a dataset comprising customer information in order to forecast if each client will make a payment call during the next 5 days. The code first loads the data into a pandas dataframe, then specifies the feature and target variables and divides the data into training and testing sets.

```
Confusion Matrix:
[[37501    98]
 [ 1360    67]]
Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98     37599
           1       0.41      0.05      0.08      1427

    accuracy                           0.96     39026
   macro avg       0.69      0.52      0.53     39026
weighted avg       0.94      0.96      0.95     39026

Accuracy Score:
0.9626402910879926
```

Code

```
# importing libraries

import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# loading the dataset

servicecallsdata = pd.read_csv('servicecallsdata.csv')

# defining the features and target variables

Features = servicecallsdata[['Tenure', 'Age', 'CHANNEL1_6M', 'CHANNEL2_6M',

'CHANNEL3_6M', 'CHANNEL4_6M', 'CHANNEL5_6M', 'METHOD1_6M',

'RECENT_PAYMENT', 'PAYMENTS_6M', 'CHANNEL1_3M', 'CHANNEL2_3M',

'CHANNEL3_3M', 'CHANNEL4_3M', 'CHANNEL5_3M', 'METHOD1_3M',

'PAYMENTS_3M', 'NOT_DI_3M', 'NOT_DI_6M', 'EVENT1_30_FLAG', 'EVENT2_90_SUM',

'LOGINS', 'POLICYPURCHASECHANNEL']]
```

```
Target = servicecallsdata['Call_Flag']
# spliting the data into training and testing sets
Features_train, Features_test, Target_train, Target_test = train_test_split(Features, Target,
test_size=0.3, random_state=0)
# scaling the feature variables using standardization
ScalingFeature = StandardScaler()
Features_train = ScalingFeature.fit_transform(Features_train)
Features_test = ScalingFeature.transform(Features_test)
#importing libraries
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
# creating a pipeline that imputes missing values and trains a logistic regression model
LogisticRegressionModel = make_pipeline(
    SimpleImputer(strategy='mean'),
    LogisticRegression(random_state=0)
)
# fiting the pipeline to the training data
LogisticRegressionModel.fit(Features_train, Target_train)
# predicting the target variable for the test set
Target_prediction = LogisticRegressionModel.predict(Features_test)
# evaluating the performance of the model
print("Confusion Matrix:")
print(confusion_matrix(Target_test, Target_prediction))
```

```
print("Classification Report:")
print(classification_report(Target_test, Target_prediction))
print("Accuracy Score:")
print(accuracy_score(Target_test, Target_prediction))
```

References

VanderPlas, J. (n.d.). *Python Data Science Handbook*. Python Data Science Handbook | Python Data Science Handbook. Retrieved May 6, 2023, from https://jakevdp.github.io/PythonDataScienceHandbook/

Wes McKinney. (n.d.). *Python for data analysis book*. Wes McKinney. Retrieved May 6, 2023, from https://wesmckinney.com/pages/book.html

Learn: Machine learning in Python. scikit. (n.d.). Retrieved May 6, 2023, from https://scikit-learn.org/stable/documentation.html

Müller Andreas C., & Guido, S. (2018). *Introduction to machine learning with python: A guide for data scientists*. O'Reilly Media, Inc.