TP3

Parcours Progis

Etudes, Medias, communication, Marketing

Bahareh Afshinpour

03.11.2025

Université Grenoble Alpes

Sciences Po Grenoble UGA

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Customer Lifetime Value (CLV)

- **CLV** is the **total amount of money** a customer is expected to spend on your business **over their entire relationship** with you.

If a customer buys a $50 product every month and stays with your brand for 2 years,
their CLV is about $1,200.

Why is CLV Important?

Marketing: Helps decide how much to spend to acquire a new customer.
Social Media: Helps identify which customers to target with ads or promotions.
Business Strategy: Helps focus on high-value customers and improve their experience.

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Step 1- Load data from a CSV file into a DataFrame

- What is a CSV file?
  - ✓A CSV (Comma-Separated Values) file is a simple text file where each line represents a data record, and each value is separated by a comma.

- Goal: Load data from a CSV file into a DataFrame, which is a table-like data structure provided by the Pandas library in Python.

Tips

- Make sure your CSV file is **in the same folder** as your Python script(code), or provide the correct path to the file.

- Check the first few rows to make sure your data was loaded correctly: **print(df.head())**

PROGIS

ETUDES, MEDIAS, COMMUNICATION, MARKETING

Step 1- Load data from a CSV file into a DataFrame

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv('./WA_Fn-UseC_-Marketing-Customer-Value-Analysis.csv', encoding='latin1')
data.columns
```

```
[3]: data.head()
```

[3]:

	Customer	State	Customer Lifetime Value	Response	Coverage	Education	Effective To Date	EmploymentStatus	Gender	Income	...
0	BU79786	Washington	2763.519279	No	Basic	Bachelor	2/24/11	Employed	F	56274	...
1	QZ44356	Arizona	6979.535903	No	Extended	Bachelor	1/31/11	Unemployed	F	0	...

Step 2- Checking for the Missing values

```
data.isnull().mean()
```

```
Customer                    0.0
State                       0.0
Customer Lifetime Value     0.0
Response                    0.0
Coverage                    0.0
Education                   0.0
Effective To Date           0.0
EmploymentStatus            0.0
Gender                      0.0
```

Step 3- Defining Independent and Target features

- Before identifying the target (dependent) and independent variables, remember:
- You should only use **numerical** data when performing regression analysis.

```
num=data.select_dtypes(include='number')
num.head()
```

	Customer Lifetime Value	Income	Monthly Premium Auto	Months Since Last Claim
0	2763.519279	56274	69	32
1	6979.535903	0	94	13
2	12887.431650	48767	108	18
3	7645.861827	0	106	18

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Step 3- Defining Independent and Target features

Independent Features :The variables used to predict the target.
Target Feature : The variable you want to predict.
Example: Independent Feature (X): Hours_Studied   Target Feature (Y): Test_Score

How to take our data frame and split it into two important parts.

Target variable

```
y = num['Customer Lifetime Value']
X = num.drop(columns=['Customer Lifetime Value'])
```

Tip: Always check the shape and content of X and y:
print(X.head())
print(y.head())

Step 3- Defining Independent and Target features

Income	Monthly Premium Auto	Customer Lifetime Value	...
56274	384.811147		
3665	1131.46493		

X

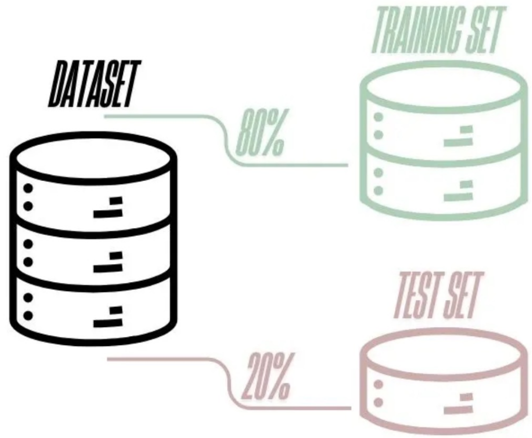
Income	Monthly Premium Auto	Total Claim Amount	...
56274	384.811147		
3665	1131.46493		

Y

Customer Lifetime Value
2763.519279
6979.535903

Step 4- Split the data into Train and Test datasets

- Why do we split the data?
  - To train the model on one part of the data (training set) and test its performance on **unseen data** (test set)
  - This helps us check if the model can generalize to new data and is not just memorizing the examples



PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Step 4- Split the data into Train and Test datasets

- The train-test split is a technique for evaluating the **performance** of a machine learning algorithm.
- It can be used for classification or regression problems and can be used for any supervised learning algorithm.
- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset**: Used to evaluate the fit machine learning model.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
```

```
# split again, and we should see the same split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
```

Example

```
>>> x
array([[ 1,  2],
       [ 3,  4],
       [ 5,  6],
       [ 7,  8],
       [ 9, 10],
       [11, 12],
       [13, 14],
       [15, 16],
       [17, 18],
       [19, 20],
       [21, 22],
       [23, 24]])
>>> y
array([0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0])
```

You probably got different results from what you see here. This is because dataset splitting is <u>random</u> by default.

```
>>> x_train, x_test, y_train, y_test = train_test_split(x, y)
>>> x_train
array([[15, 16],
       [21, 22],
       [11, 12],
       [17, 18],
       [13, 14],
       [ 9, 10],
       [ 1,  2],
       [ 3,  4],
       [19, 20]])
>>> x_test
array([[ 5,  6],
       [ 7,  8],
       [23, 24]])
>>> y_train
array([1, 1, 0, 1, 0, 1, 0, 1, 0])
>>> y_test
array([1, 0, 0])
```

PROGIS

ETUDES, MEDIAS, COMMUNICATION, MARKETING

Step 4- Split the data into Train and Test datasets

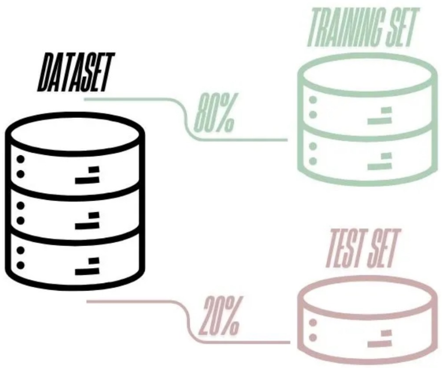
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print("shape of X_train ",X_train.shape)
print("shape of X_test ",X_test.shape)
```

```
shape of X_train  (7307, 7)
shape of X_test  (1827, 7)
```

Every time we change the random state, different observation gets selected into the training and testing.

If the averaging value of the dependent variable differs significantly between training and testing, the model does not have a fair opportunity of learning what it can from the data.



Step 5- Fitting the simple regression model

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train, y_train)
print (reg.intercept_)
#linear_predict_all=reg.predict(X)
```

```
reg.intercept_
```

```
193.6434424279978
```

```
reg.coef_
```

Y=a*x+b

Y=coef*x+intercept

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Step 5- Linear Regression

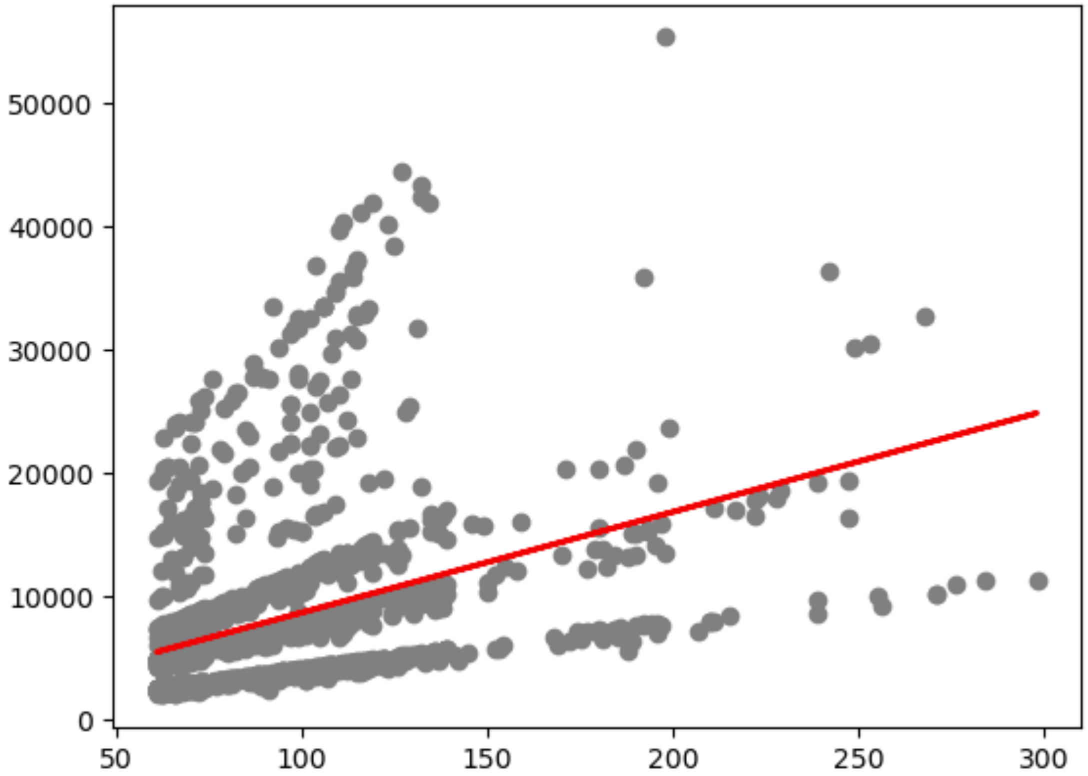
```
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(X_train, y_train)
acc_log_train = round(reg.score(X_train, y_train)*100,2)
acc_log_test = round(reg.score(X_test, y_test)*100,2)
print("Training Accuracy:%{}".format(acc_log_train))
print("Testing Accuracy:%{}".format(acc_log_test))
```

```
Training Accuracy:%16.2
Testing Accuracy:%15.3
```

It is not good

```
test_p = reg.predict(X_test)

from sklearn.metrics import r2_score
print('R-Squared for Test set: %0.2f' % r2_score(y_true=y_test, y_pred=test_p))
```



It is not good

We can do

Linear Regression By filtering with 'Number of Policies'

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Why Not All Columns Improve Regression?

- When building a regression model to predict something like Customer Lifetime Value, it's tempting to include all available columns (features) in your dataset.
- However, using every column doesn't always lead to a better model. **Sometimes, it can even make your predictions worse**

Not All Features Are Useful:

- Some columns have little or no relationship with the target variable (Customer Lifetime Value).
- Including unrelated or random columns can introduce noise and confuse the model.
- Unnecessary features can "drown out" the effect of important ones, making it harder for the model to learn real relationships.

Try Feature Selection Techniques

- Factors positively influencing CLV include Monthly Premium and Number of Policies, while Open Complaints and Claim Amount have the potential to decrease CLV. These factors should be carefully managed to optimize customer value.

FUNCTION

There are two kinds of functions in Python.

-Built-in functions that are provided as part of Python

- print(), input(), type(), float(), int() ...

- Functions that we define ourselves and then use

FUNCTION

-In Python a function is some **reusable** code that takes arguments(s) as input, does some computation, and then returns a result or results

-We define a function using the **def** reserved word

-We **call/invoke** the function by using the function name, parentheses, and arguments in an expression

Function example

Creating a Function

```
def my_function():
  print("Hello from a function")
```

Calling a Function

```
.....
....
my_function()
```

Arguments

- An argument is a value we pass into the function as its input when we call the function

Parameters

A parameter is a variable which we use in the function definition. It is a "handle" that allows the code in the function to access the arguments for a particular function invocation.

```
>>> greet('en')
Hello
>>> greet('fr')
Bonjour
>>>
```

Argument

Parameter

```
>>> def greet(lang):
...     if lang == 'en':
...        print('Hello')
...     elif lang == 'fr':
...        print('Bonjour')
```

Return Values

```
def my_function(x):
  return 5 * x



print(my_function(3))
print(my_function(5))
print(my_function(9))
```

End