About the algorithm

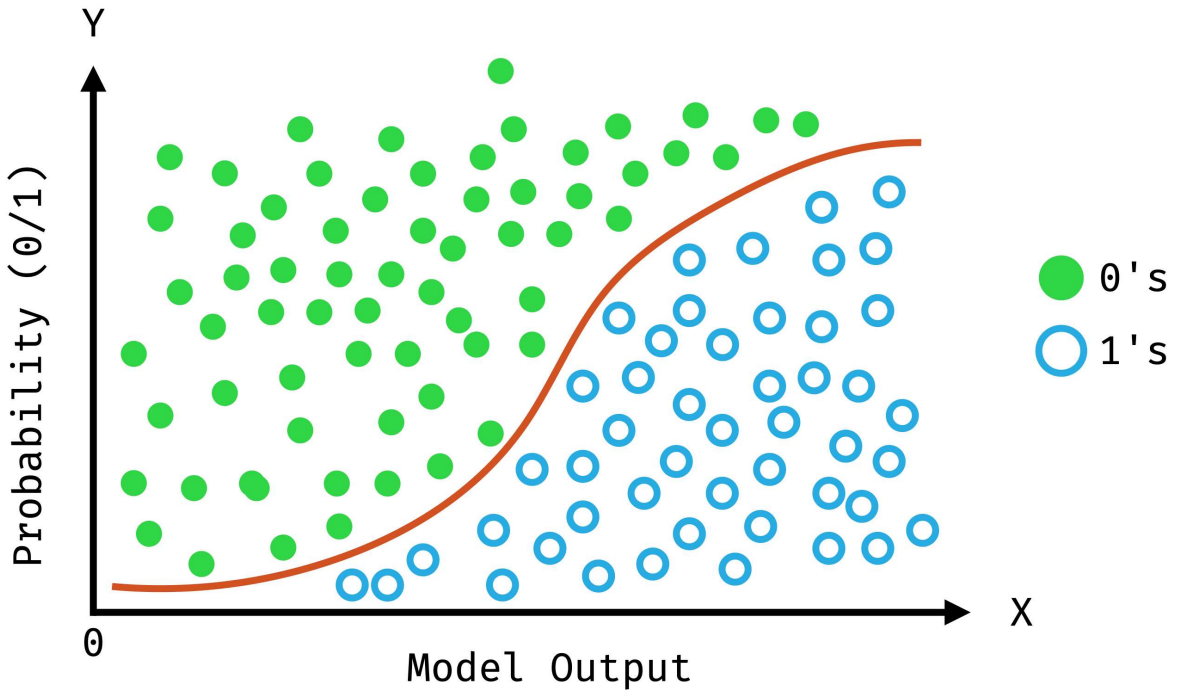
- Logistic regression is a binary classification technique that predicts the probability of an observation belonging to a particular category. Key points:

1. **Binary Outcome**: Used for binary outcomes, with the dependent variable having two categories.
2. **Logit Function**: Utilizes the logistic function to model the relationship between independent variables and the log-odds of the outcome, ensuring predicted probabilities fall between 0 and 1.
3. **Model Training**: Coefficients are estimated via techniques like maximum likelihood estimation to optimize the likelihood of observing actual outcomes given predictor variables.
4. **Decision Boundary**: Establishes a boundary based on learned coefficients, dividing the space into regions corresponding to the two outcomes.
5. **Prediction**: After training, the model calculates probabilities for each category, with a threshold (often 0.5) applied to classify observations.

Logistic regression is widely applied across domains due to its simplicity, interpretability, and efficacy in binary classification tasks.

In [1]:
```
from IPython.display import Image
image_path = "4.regression.003.jpeg"
Image(filename=image_path)
```

Out[1]:



✔✨ Import Libraries

In [114…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
from IPython.display import display
import warnings
```

✔✨ Loading data for binary classification

In [115…
```
data = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")
data.head()
```

Out[115]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	De
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	

5 rows × 21 columns

✔✨ Data Cleaning

In [116…
```
data.isnull().sum()
```

Out[116]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

✨convert Totalcharges from object to numeric if possible , else set it as Null

In [117…
```
data["TotalCharges"] = pd.to_numeric(data["TotalCharges"] ,errors="coerce")
```

In [118…
```
data.isnull().sum()
```

Out[118]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges       11
Churn               0
dtype: int64
```

 - Note that null values are in Totalcharges column, so we need to handle it

 Fill null values with mean

In [119…
```
data.fillna(data["TotalCharges"].mean() ,inplace=True)
```

In [120…
```
data.isnull().sum()
```

Out[120]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

In [121…
```
data.columns = data.columns.str.lower().str.replace(" ","_")
```

In [122…
```
for col in data.columns:
    if data[col].dtype =="object":
        data[col]  = data[col].str.lower().str.replace(" ","_")
```

In [123…
```
data["seniorcitizen"] = data["seniorcitizen"].astype("object")
```

✔✨ Data Analysis

In [124…
```
data.describe()
```

Out[124]:

	tenure	monthlycharges	totalcharges
count	7043.000000	7043.000000	7043.000000
mean	32.371149	64.761692	2283.300441
std	24.559481	30.090047	2265.000258
min	0.000000	18.250000	18.800000
25%	9.000000	35.500000	402.225000
50%	29.000000	70.350000	1400.550000
75%	55.000000	89.850000	3786.600000
max	72.000000	118.750000	8684.800000

In [125…

```
(data==0).sum()
```

Out[125]:
```
customerid            0
gender                0
seniorcitizen      5901
partner               0
dependents            0
tenure               11
phoneservice          0
multiplelines         0
internetservice       0
onlinesecurity        0
onlinebackup          0
deviceprotection      0
techsupport           0
streamingtv           0
streamingmovies       0
contract              0
paperlessbilling      0
paymentmethod         0
monthlycharges        0
totalcharges          0
churn                 0
dtype: int64
```

In [126…
```
data = data[data["tenure"]!=0]
(data==0).sum()
```

Out[126]:
```
customerid            0
gender                0
seniorcitizen      5890
partner               0
dependents            0
tenure                0
phoneservice          0
multiplelines         0
internetservice       0
onlinesecurity        0
onlinebackup          0
deviceprotection      0
techsupport           0
streamingtv           0
streamingmovies       0
contract              0
paperlessbilling      0
paymentmethod         0
monthlycharges        0
totalcharges          0
churn                 0
dtype: int64
```

✨ Describe categorical data

In [127…
```
data.describe(include="object")
```

Out[127]:

	customerid	gender	seniorcitizen	partner	dependents	phoneservice	multiplelines	internetservice	onlinesecurity	onlinebacku
count	7032	7032	7032	7032	7032	7032	7032	7032	7032	703
unique	7032	2	2	2	2	2	3	3	3	
top	7590-vhveg	male	0	no	no	yes	no	fiber_optic	no	n
freq	1	3549	5890	3639	4933	6352	3385	3096	3497	308

In [128…
```
data['churn'].value_counts()
1869/7043
```

Out[128]:
```
0.2653698707936959
```

In [129…
```
data['churn']=(data['churn']=="yes").astype("int64")
```

In [130…
```
mean = data["churn"].mean()   #know mean
```

In [131…
```
data["gender"].value_counts()
```

Out[131]:
```
male      3549
female    3483
Name: gender, dtype: int64
```

In [132…
```
males = data[data["gender"] =="male"]
males.churn.mean()
```

Out[132]:
```
0.26204564666103125
```

In [133…
```
females = data[data["gender"] =="female"]
females.churn.mean()
```

Out[133]:
```
0.2695951765719208
```

- 🕵️‍♀️ Note : (gender) :The average of men who unsubscribed is the same as women, so this feature has no effect

In [134…
```
senior1 = data[data["seniorcitizen"] == 0]
senior1.churn.mean()
```

Out[134]:
```
0.2365025466893039
```

- 🕵️‍♀️ Note :
  - (seniorcitizen) :The largest percentage cancels, so this feature is important

In [135…
```
senior2 = data[data["seniorcitizen"] == 1]
senior2.churn.mean()
```

Out[135]:
```
0.4168126094570928
```

✔✨ Do the same with other categorical features

In [136…
```
categorical = data.select_dtypes("object").columns
categorical = categorical.drop("customerid")            # drop id  column from categorical features
categorical
```

Out[136]:
```
Index(['gender', 'seniorcitizen', 'partner', 'dependents', 'phoneservice',
       'multiplelines', 'internetservice', 'onlinesecurity', 'onlinebackup',
       'deviceprotection', 'techsupport', 'streamingtv', 'streamingmovies',
       'contract', 'paperlessbilling', 'paymentmethod'],
      dtype='object')
```

In [137…
```
for col in categorical:
    df_groups = data.groupby(col).churn.agg(["mean"])
    df_groups["diff"] = (df_groups["mean"] - mean )        # feature mean - target mean
    display(df_groups)
```

gender	mean	diff
female	0.269595	0.003810
male	0.262046	-0.003739

seniorcitizen	mean	diff
0	0.236503	-0.029282
1	0.416813	0.151028

partner	mean	diff
no	0.329761	0.063976
yes	0.197171	-0.068614

dependents	mean	diff
no	0.312791	0.047006
yes	0.155312	-0.110473

phoneservice	mean	diff
no	0.250000	-0.015785
yes	0.267475	0.001690

multiplelines	mean	diff
no	0.250812	-0.014973
no_phone_service	0.250000	-0.015785
yes	0.286485	0.020700

internetservice	mean	diff
dsl	0.189983	-0.075802
fiber_optic	0.418928	0.153143
no	0.074342	-0.191443

onlinesecurity	mean	diff
no	0.417787	0.152002
no_internet_service	0.074342	-0.191443
yes	0.146402	-0.119383

onlinebackup	mean	diff
no	0.399417	0.133632
no_internet_service	0.074342	-0.191443
yes	0.215670	-0.050115

deviceprotection	mean	diff
no	0.391403	0.125618
no_internet_service	0.074342	-0.191443
yes	0.225393	-0.040392

techsupport	mean	diff
no	0.416475	0.150690
no_internet_service	0.074342	-0.191443
yes	0.151961	-0.113824

streamingtv	mean	diff
no	0.335351	0.069566
no_internet_service	0.074342	-0.191443
yes	0.301147	0.035362

streamingmovies	mean	diff
no	0.337289	0.071504
no_internet_service	0.074342	-0.191443
yes	0.299524	0.033739

contract	mean	diff
month-to-month	0.427097	0.161312
one_year	0.112772	-0.153013
two_year	0.028487	-0.237298

paperlessbilling	mean	diff
no	0.163757	-0.102028
yes	0.335893	0.070108

paymentmethod	mean	diff
bank_transfer_(automatic)	0.167315	-0.098470
credit_card_(automatic)	0.152531	-0.113254
electronic_check	0.452854	0.187069
mailed_check	0.192020	-0.073765

- 🕵️ Note : drop features with small difference (have no effect)

In [138…
```
data.drop(["customerid", "gender","dependents","phoneservice","multiplelines","streamingmovies"], axis=1, inplace=T
```

✨ Correlation

In [139…
```
plt.figure(figsize=(15,6))
sns.heatmap(data.corr(),annot=True)
```

```
warnings.filterwarnings("ignore", message="The default value of numeric_only in DataFrame.corr is deprecated.", cat
warnings.filterwarnings("ignore", message="X has feature names, but LogisticRegression was fitted without feature n
```

🦸 - Note : there is strong relationship between (total charges) and (monthlycharges) >> so we will drop one of them which has weaker relationship with tatget ... drop (totalcharges)

In [140…

```
data.drop(["totalcharges"], axis=1, inplace=True)
```

✨ Encoding categorical data

In [143…

```
categorical = data.select_dtypes("object").columns
data = pd.get_dummies(data, columns= categorical)
```

```
warnings.filterwarnings("ignore", message="In a future version, the Index constructor will not infer numeric dtypes
```

✨ Split data (x , y)

In [144…
```
x = data.drop(["churn"], axis=1)
y = data["churn"]
```

✨Split x (train and test)

In [145…
```
x_train , x_test, y_train, y_test = train_test_split(x, y, test_size=.2, random_state=0)
```

✨ Scalling

In [146…
```
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x_train)
```

✨ Build Model

In [147…
```
model = LogisticRegression(solver="saga",penalty='l1')
model.fit(x_scaled , y_train)
```

Out[147]:
```
LogisticRegression
LogisticRegression(penalty='l1', solver='saga')
```

✨ Check accurecy

In [148…
```
model.score(x_scaled, y_train)*100
```

Out[148]:
```
80.37333333333333
```

In [149…
```
model.score(x_test, y_test)*100
```

Out[149]:
```
77.46979388770433
```

✨ Predict probability >> both : p(y=1) , p(y=0)

In [150…
```
y_predicted = model.predict_proba(x_test)
y_predicted
```

Out[150]:
```
array([[2.99056091e-01, 7.00943909e-01],
       [9.99983146e-01, 1.68542024e-05],
       [3.82309039e-01, 6.17690961e-01],
       ...,
       [4.02197202e-03, 9.95978028e-01],
       [1.00000000e+00, 6.02661158e-16],
       [7.28101091e-01, 2.71898909e-01]])
```

✨ Predict probability >> set threshold and Predict if the output belongs to class (1) or not

In [151…
```
y_predicted = y_predicted[:, 1]
y_predicted = (y_predicted > .7).astype('int64')  #threshold
y_predicted
```

Out[151]:
```
array([1, 0, 0, ..., 1, 0, 0], dtype=int64)
```

✨ Predict if the output belongs to class (0) or class (1) with default threshold of 0.5

In [152…
```
y = model.predict(x_test)
y
```

Out[152]:
```
array([1, 0, 1, ..., 1, 0, 0], dtype=int64)
```

Thanks 💔