Introduction to Machine Learning

Dr. Ernesto Lee

Welcome to the Class

What we are going to learn

- What is machine learning?

  a) Is machine learning hard? (Spoiler: No)

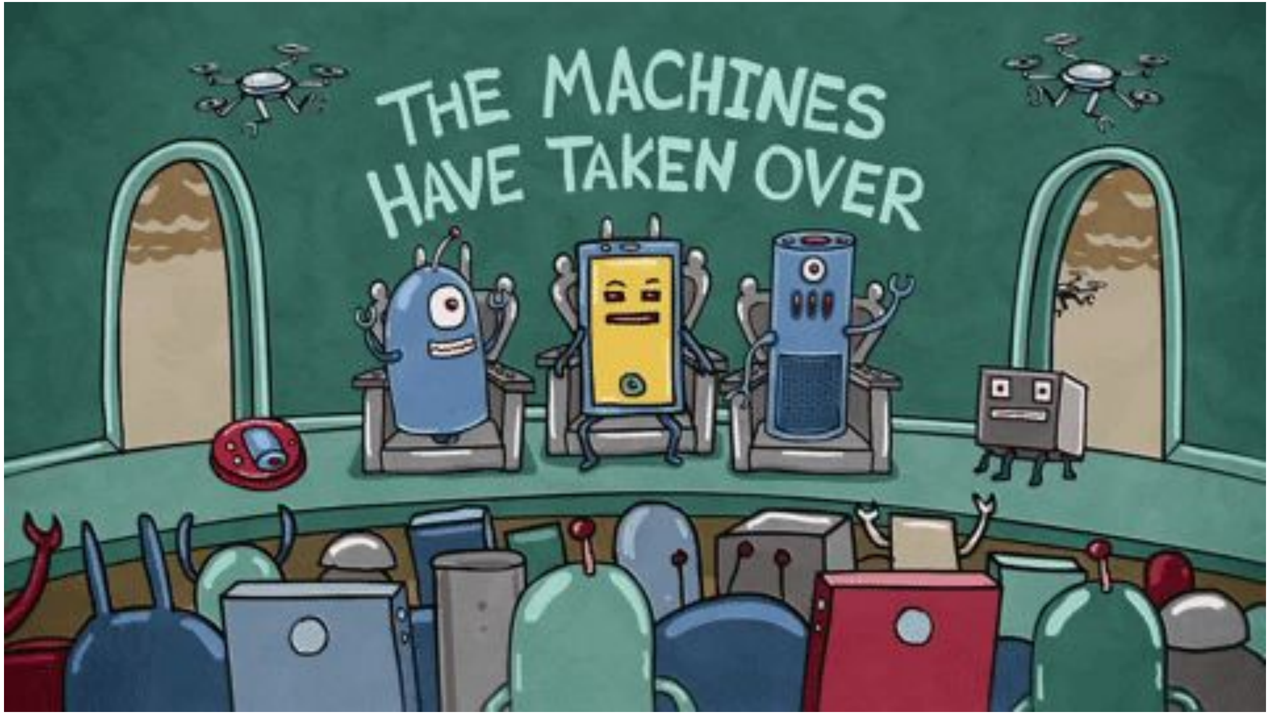
  b) What do we learn in this course?

  c) What is artificial intelligence and how does it differ from machine learning?

  d) How do humans think, and how can we inject those ideas into a machine?

- Some basic machine learning examples in real life.

Machine Learning is Everywhere



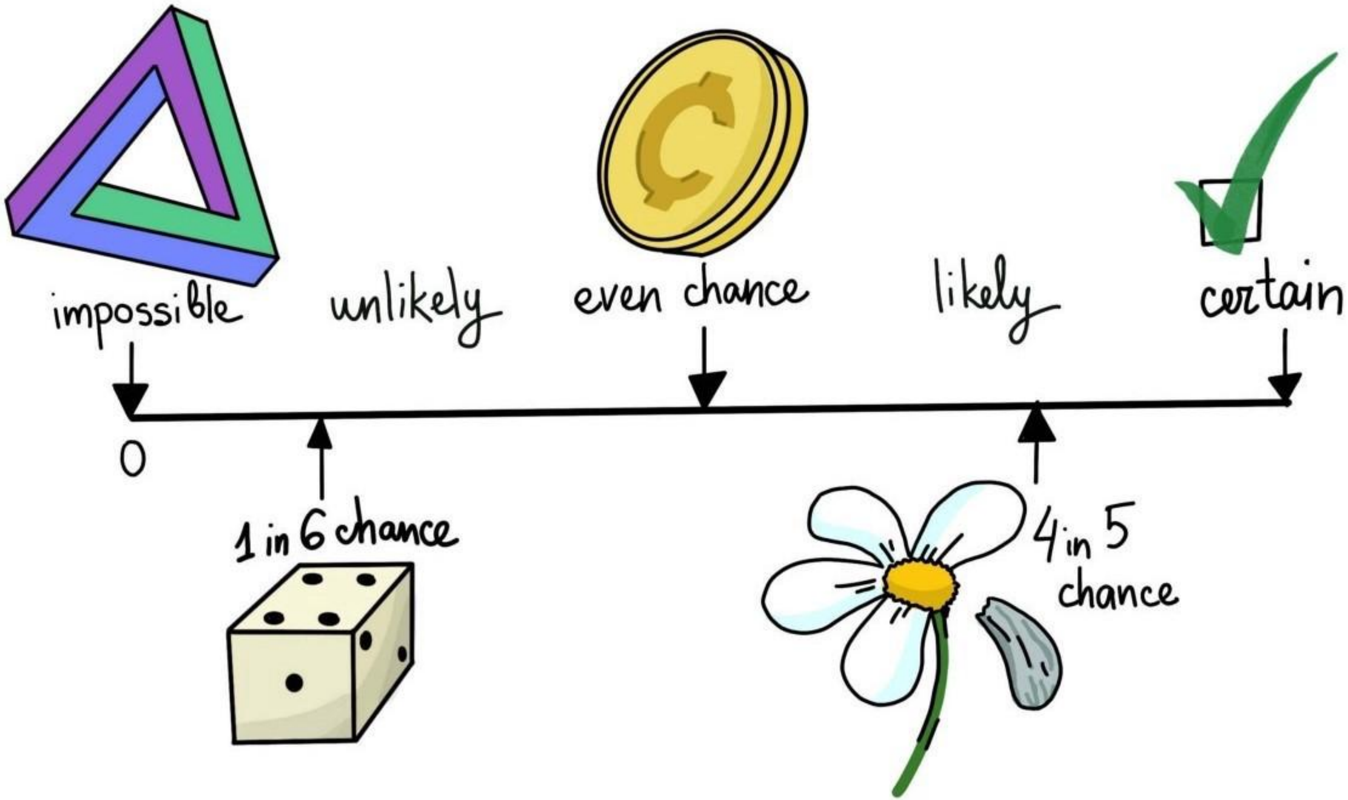
Do I need Heavy Math and Coding to Understand?

No.

Formulas and Python are a Language

$$\sum_{i=1}^{100} i.$$

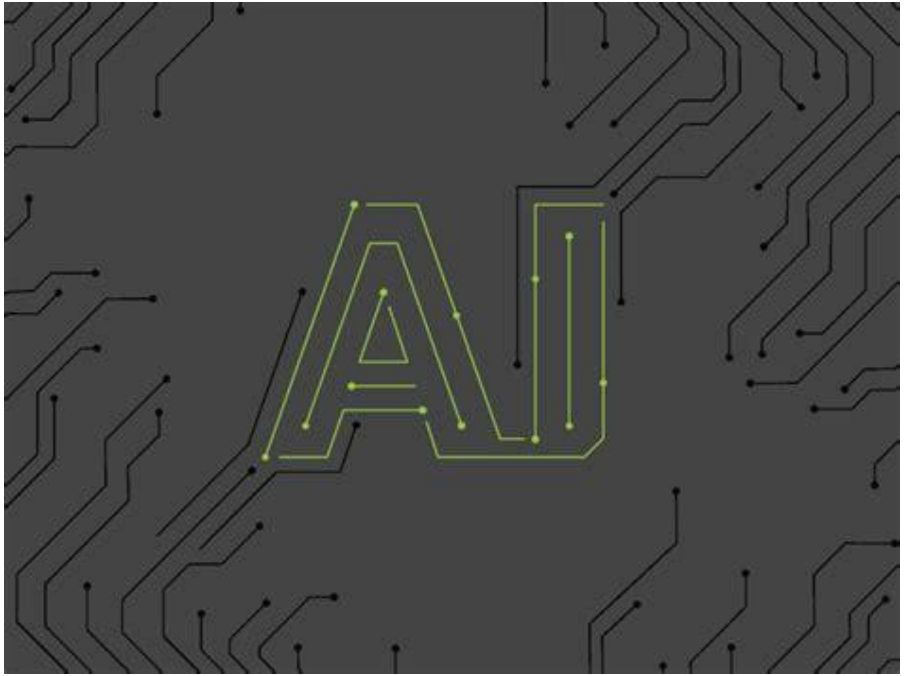
Probability



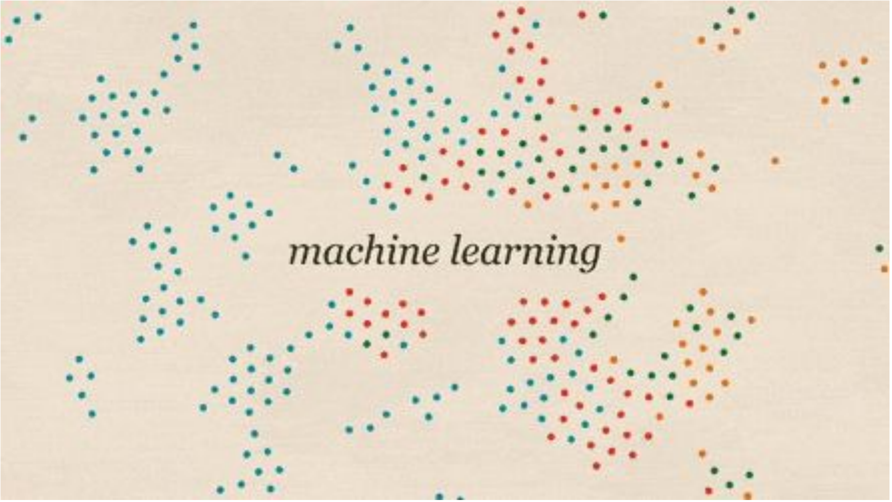
What is Machine Learning

What is AI?

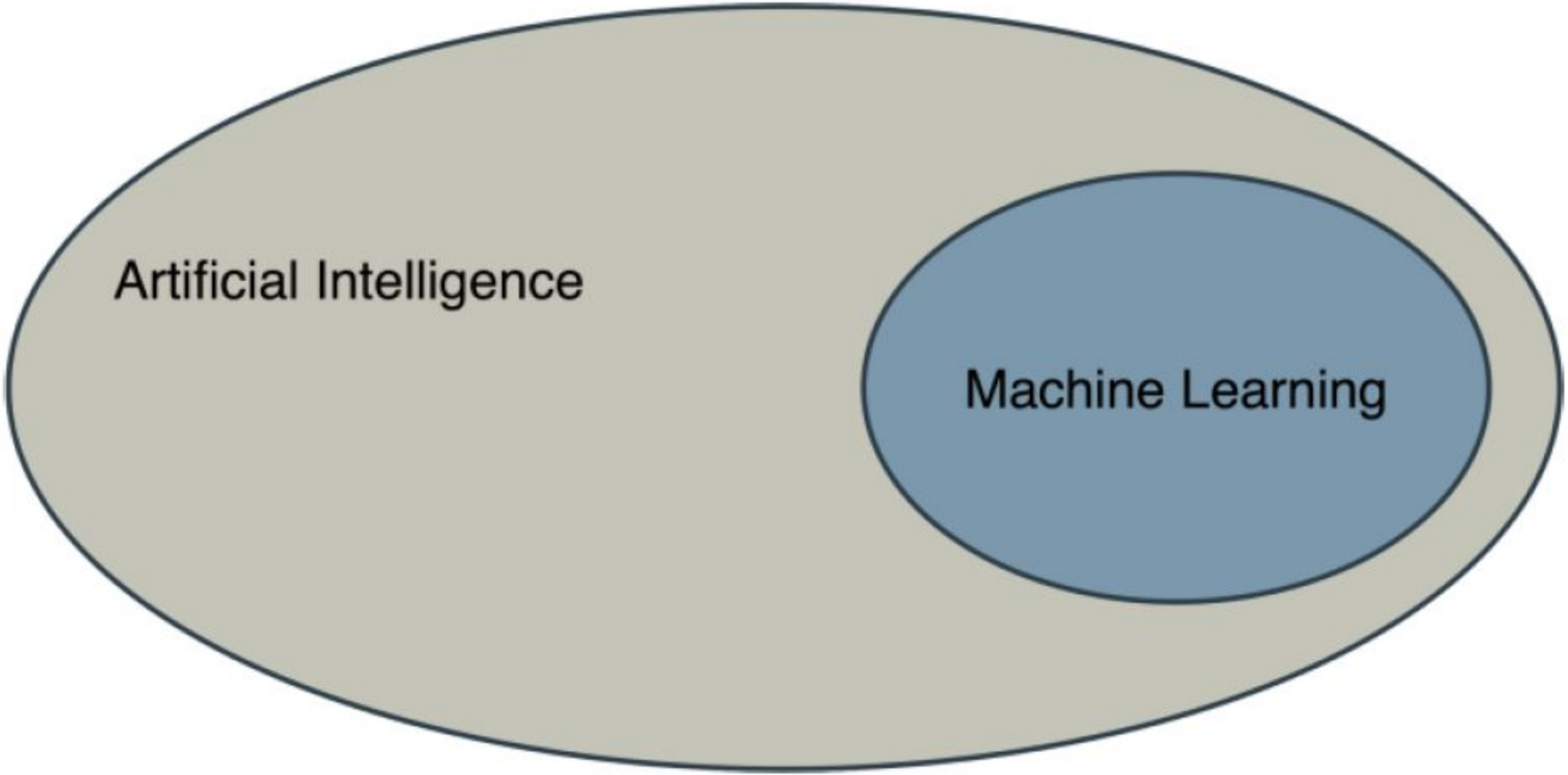
AI is when a computer mimics human behavior.



What is Machine Learning?



MACHINE LEARNING The set of all tasks in which a computer can make decisions based on data.


Artificial Intelligence
Machine Learning

What is Machine Learning?

Machine Learning is COMMON SENSE… except done by a computer.

Machine learning encompasses all the tasks in which computers make decisions based on data.

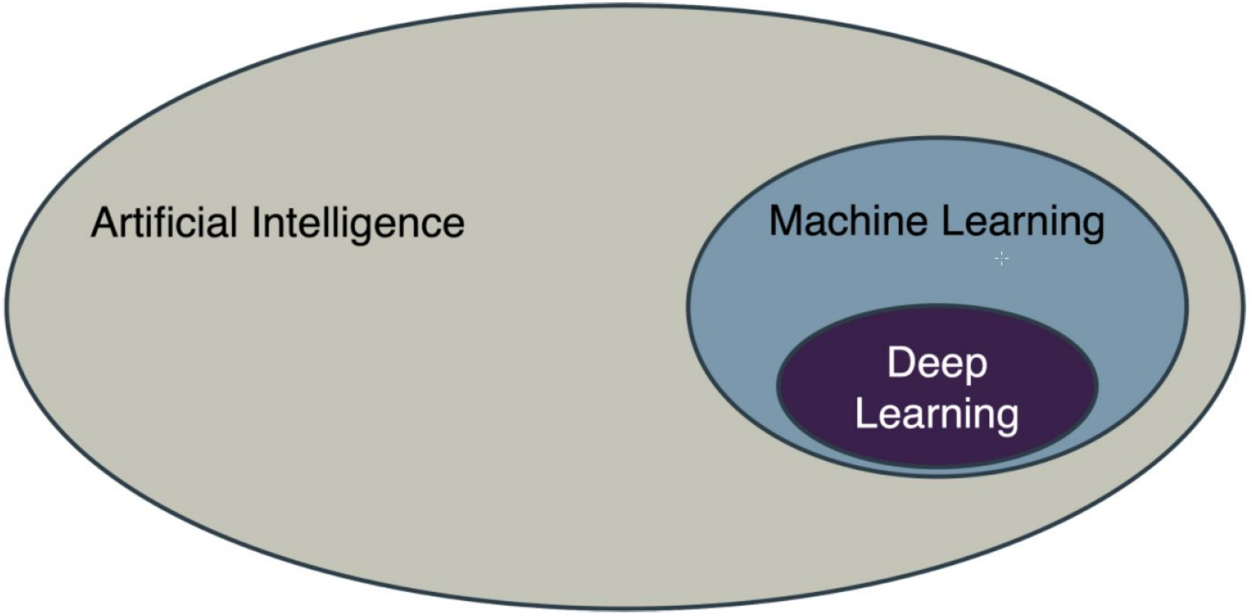
In the same way that humans make decisions based on previous experiences, computers can make decisions based on previous data.

What is Machine Learning?

Machine Learning is a LABEL MAKER.

What is Deep Learning?

DEEP LEARNING: The field of machine learning that uses certain objects called neural networks.



Remember, Formulate, Predict

1. We remember past situations that were similar.

2. We formulate a general rule.

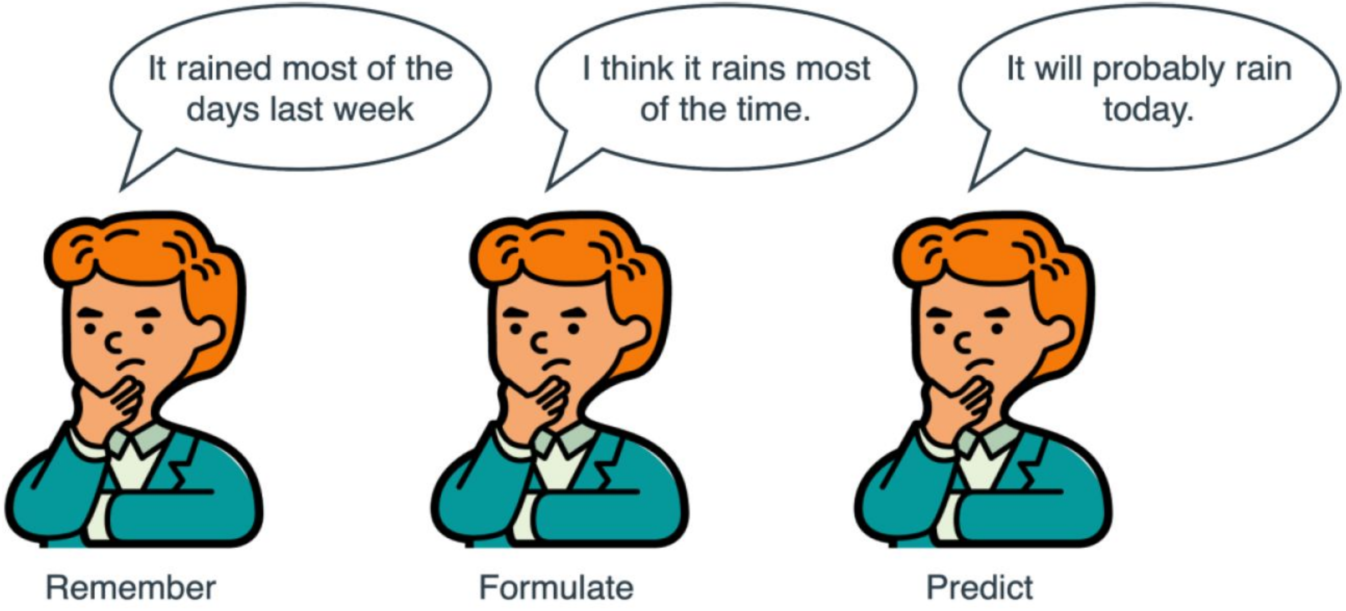
3. We use this rule to predict what may happen in the future.

How do Humans Think?

1. We remember that last week it rained most of the time.

2. We formulate that in this place, it rains most of the time.

3. We predict that today it will rain.



Model versus and Algorithm

MODEL: A set of rules that represent our data and can be used to make predictions.

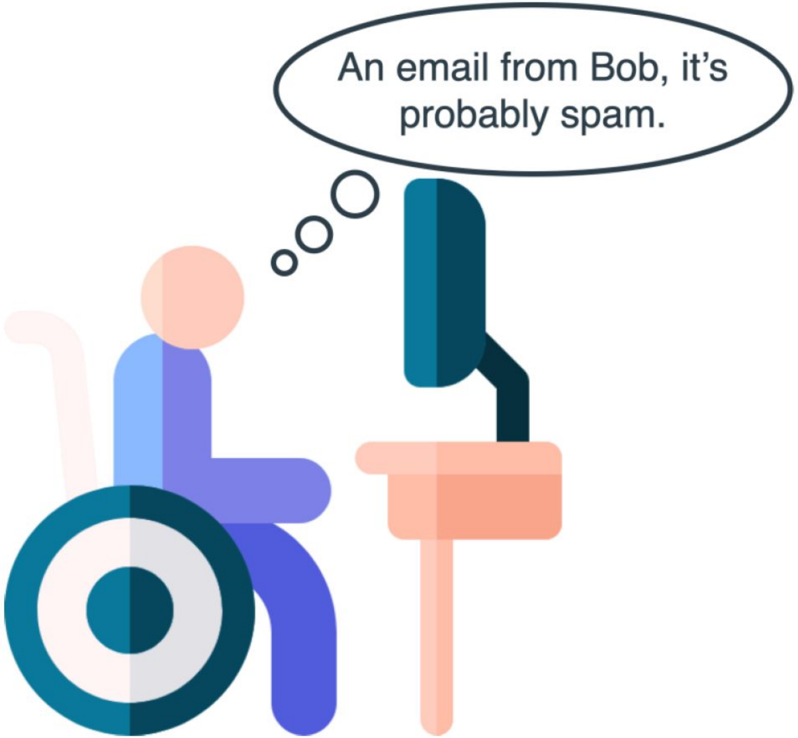
ALGORITHM: A procedure, or a set of steps, used to solve a problem or perform a computation. In this course, the goal of an algorithm is to build a model.

EMAIL EXAMPLE

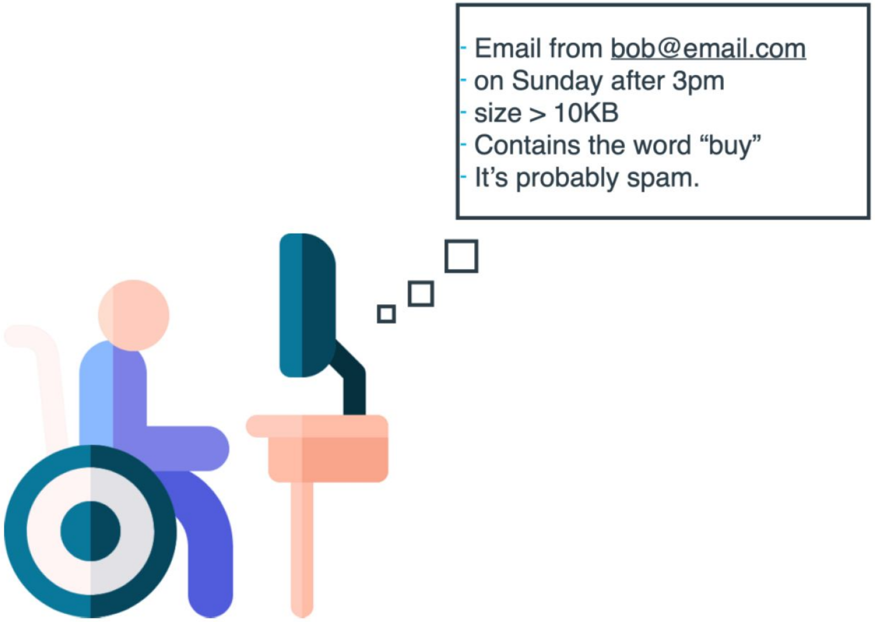
The last 10 emails that we got from Bob. That is our data. We **remember** that 4 of them were spam and the other 6 were ham. From this information, we can **formulate** the following model:

<u>Model 1: 4 out of every 10 emails that Bob sends us are spam.</u>

We **predict** that 60% of emails are ham.



Email Example



(size) + 10(number of spelling mistakes) - (number of appearances of the word ‘mom’) +

4(number of appearances of the word ‘buy’) > 10,

then we classify the message as spam. Otherwise we classify it as ham.

Summary

o Remember: Look at previous data.

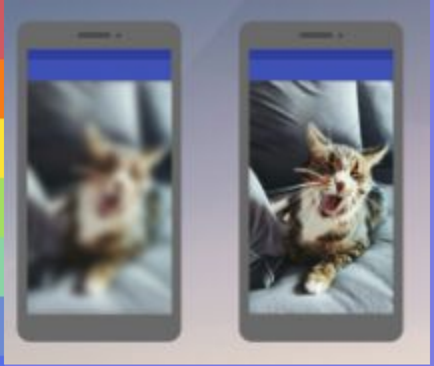
o Formulate: Build a model, or a rule, based on this data.

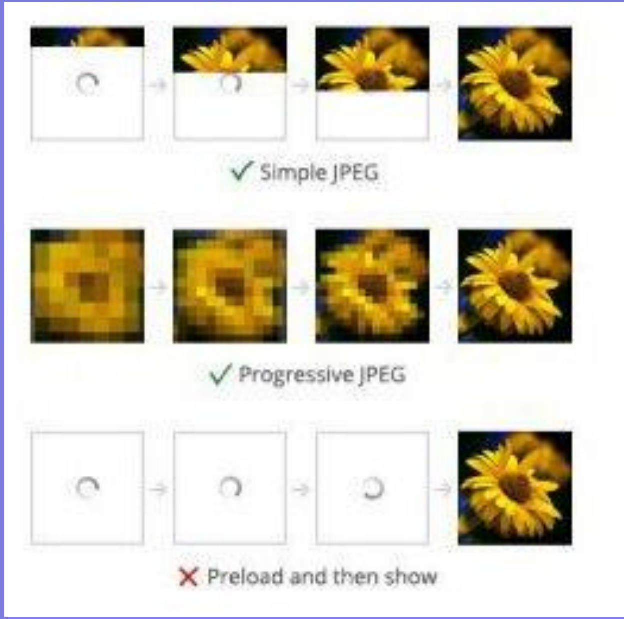
o Predict: Use the model to make predictions about future data.

Data Science Demo

Dr. Ernesto Lee

Why This is Different





ADEPT Method for Learning	
Analogy	Tell me what it's like.
Diagram	Help me visualize it.
Example	Allow me to experience it.
Plain English	Describe it with everyday words.
Technical Definition	Discuss the formal details.

Data

https://raw.githubusercontent.com/fenago/pythonml/main/data/WA_Fn-UseC_-Telco-Customer-Churn.csv

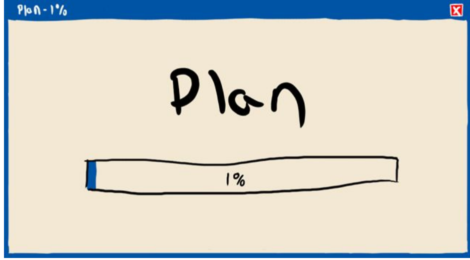
What you will Learn

- Doing exploratory data analysis for identifying important features
- Encoding categorical variables to use them in machine learning models
- Using logistic regression for classification

Churn Project

Are customers going to leave the company?

The Plan for the Project



1. First, we download the dataset and do some initial preparation: rename columns and change values inside columns to be consistent throughout the entire dataset.
2. Then we split the data into train, validation, and test so we can validate our models.
3. As part of the initial data analysis, we look at feature importance to identify which features are important in our data.
4. We transform categorical variables into numeric so we can use them in the model.
5. Finally, we train a logistic regression model.

The Dataset

- **Services of the customers** – phone; multiple lines; internet; tech support and extra services such as online security, backup, device protection, and TV streaming
- **Account information** – how long they have been clients, type of contract, type of payment method
- **Charges** – how much the client was charged in the past month and in total
- **Demographic information** – gender, age, and whether they have dependents or a partner
- **Churn** – yes/no, whether the customer left the company within the past month

https://www.kaggle.com/blastchar/telco-customer-churn

Initial data preparation

Do your Imports

```
import pandas as pd

import numpy as np


import seaborn as sns

from matplotlib import pyplot as plt

%matplotlib inline
```

Read the Dataset

```
df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')

len(df)
```

```
df.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...
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	...

Transpose the Dataset to Make it Wide (not long)

```
df.head().T
```

	0	1	2
customerID	7590-VHVEG	5575-GNVDE	3668-QPYBK
gender	Female	Male	Male
SeniorCitizen	0	0	0
Partner	Yes	No	No
Dependents	No	No	No
tenure	1	34	2
PhoneService	No	Yes	Yes
MultipleLines	No phone service	No	No
InternetService	DSL	DSL	DSL
OnlineSecurity	No	Yes	Yes
OnlineBackup	Yes	No	Yes
DeviceProtection	No	Yes	No
TechSupport	No	No	No
StreamingTV	No	No	No
StreamingMovies	No	No	No
Contract	Month-to-month	One year	Month-to-month
PaperlessBilling	Yes	No	Yes
PaymentMethod	Electronic check	Mailed check	Mailed check
MonthlyCharges	29.85	56.95	53.85
TotalCharges	29.85	1889.5	108.15
Churn	No	No	Yes

Data Types

```
df.dtypes
```

```
total_charges = 
pd.to_numeric(df.TotalCharges, 
errors='coerce')
```

```
df.dtypes
```

```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

SeniorCitizen is integer

TotalCharges is not correctly identified as a numeric type (float or int)

Consider Empties...

```
df[total_charges.isnull()][['customerID', 'TotalCharges']]

df.TotalCharges = pd.to_numeric(df.TotalCharges,
errors='coerce')

df.TotalCharges = df.TotalCharges.fillna(0)
```

Column Names and Naming Conventions

```
df.columns = df.columns.str.lower().str.replace(' ', '_')

string_columns = list(df.dtypes[df.dtypes ==
'object'].index)

for col in string_columns:

    df[col] = df[col].str.lower().str.replace(' ', '_')
```

Encode Churn Target Variable

```
df.churn = (df.churn == 'yes').astype(int)
```

Split the Data for Testing and Training

```
from sklearn.model_selection import train_test_split

df_train_full, df_test = train_test_split(df, test_size=0.2,
random_state=1)
```

	customerid	gender	tenure
0	7590-vhveg	female	1
1	5575-gnvde	male	34
2	3668-qpybk	male	2
3	7795-cfocw	male	45
4	9237-hqitu	female	2
5	9305-cdskc	female	8
6	1452-kiovk	male	22
7	6713-okomc	female	10
8	7892-pookp	female	28
9	6388-tabgu	male	62

Shuffle

	customerid	gender	tenure
8	7892-pookp	female	28
2	3668-qpybk	male	2
5	9305-cdskc	female	8
6	1452-kiovk	male	22
3	7795-cfocw	male	45
1	5575-gnvde	male	34
0	7590-vhveg	female	1
7	6713-okomc	female	10
4	9237-hqitu	female	2
9	6388-tabgu	male	62

Split 80%/20%

Train

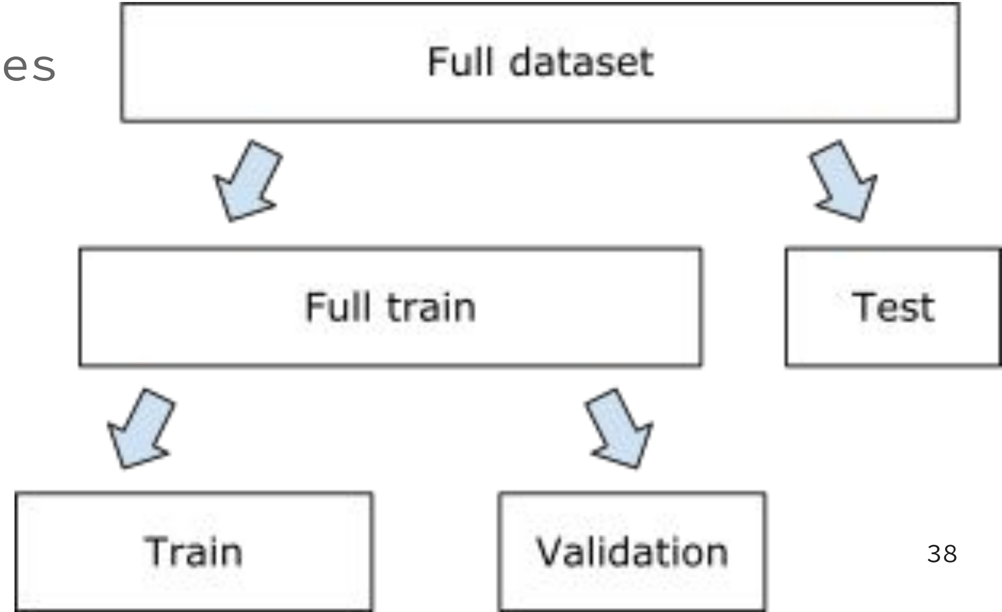
	customerid	gender	tenure
8	7892-pookp	female	28
2	3668-qpybk	male	2
5	9305-cdskc	female	8
6	1452-kiovk	male	22
3	7795-cfocw	male	45
1	5575-gnvde	male	34
0	7590-vhveg	female	1
7	6713-okomc	female	10

	customerid	gender	tenure
4	9237-hqitu	female	2
9	6388-tabgu	male	62

Test

Train, Test, Validate

```
df_train, df_val = train_test_split(df_train_full,
test_size=0.33, random_state=11)

y_train = df_train.churn.values

y_val = df_val.churn.values

del df_train['churn']

del df_val['churn']
```



Exploratory Data Analysis

Validate There are No Missing Values

```
df_train_full.isnull().sum()
```

```
df_train_full.isnull().sum()
```

```
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
```

Validate the Distribution of the Target Variable

```
df_train_full.churn.value_counts()
```

What percentage of customers *STOPPED* using the services?

Compute the MEAN of the Target Variable

```
global_mean = df_train_full.churn.mean()
```

```
global_mean = df_train_full.churn.mean()
round(global_mean, 3)
```

```
0.27
```

Imbalanced DAtaset

Categorical & Numerical Columns Require Different Treatments

- **categorical**, which will contain the names of categorical variables,
- **numerical**, will have the names of numerical variables

```
categorical = ['gender', 'seniorcitizen', 'partner', 'dependents',
               'phoneservice', 'multiplelines', 'internetservice',
               'onlinesecurity', 'onlinebackup', 'deviceprotection',
               'techsupport', 'streamingtv', 'streamingmovies',
               'contract', 'paperlessbilling', 'paymentmethod']
numerical = ['tenure', 'monthlycharges', 'totalcharges']
```

Categorical Data

```
df_train_full[categorical].nunique()
```

```
df_train_full[categorical].nunique()
```

```
gender              2
seniorcitizen       2
partner             2
dependents          2
phoneservice        2
multiplelines       3
internetservice     3
onlinesecurity      3
onlinebackup        3
deviceprotection    3
techsupport         3
streamingtv         3
streamingmovies     3
contract            3
paperlessbilling    2
paymentmethod       4
dtype: int64
```

Numerical Data

Get the Descriptive statistics for each column (Univariate Analysis)

```
df_train_full[numerical].describe()
```

Correlations

```
df_train_full.corr()
```

Feature Importance

Feature Importance

- Knowing how other variables affect the target variable, churn, is the key to understanding the data and building a good model.
  - This process is called **feature importance** analysis
- We have two different kinds of features: categorical and numerical.
  - Each kind has different ways of measuring feature importance, so we will look at each separately.

Feature Importance: Categorical Variables

	customerid	gender	churn
0	7590-vhveg	female	0
1	5575-gnvde	male	0
2	3668-qpybk	male	1
3	7795-cfocw	male	0
4	9237-hqitu	female	1
5	9305-cdskc	female	1
6	1452-kiovk	male	0
7	6713-okomc	female	0
8	7892-pookp	female	1
9	6388-tabgu	male	0

`gender == "female"`

	customerid	gender	churn
0	7590-vhveg	female	0
4	9237-hqitu	female	1
5	9305-cdskc	female	1
7	6713-okomc	female	0
8	7892-pookp	female	1

	customerid	gender	churn
1	5575-gnvde	male	0
2	3668-qpybk	male	1
3	7795-cfocw	male	0
6	1452-kiovk	male	0
9	6388-tabgu	male	0

`gender == "male"`

Strategy for Categorical Feature Importance

- So we can look at all the distinct values of a variable.
  - for each variable, there's a group of customers: all the customers who have this value.
  - For each such group, we can compute the churn rate, which will be the group churn rate.
  - When we have it, we can compare it with the global churn rate — churn rate calculated for all the observations at once.
- If the difference between the rates is small, the value is not important when predicting churn because this group of customers is not really different from the rest of the customers.
- On the other hand, if the difference is not small, something inside that group sets it apart from the rest.

An Analyst or machine learning algorithm should be able to pick this up and use it when making predictions.

Feature Importance Based on Gender

Let’s check first for the gender variable. To compute the churn rate for all female customers, we first select only rows that correspond to gender == 'female' and then compute the churn rate for them:

```
female_mean = df_train_full[df_train_full.gender == 'female'].churn.mean()
```

```
male_mean = df_train_full[df_train_full.gender == 'male'].churn.mean()
```

Feature Importance based on Partner

partner_yes = df_train_full[df_train_full.partner == 'yes'].churn.mean()

partner_no = df_train_full[df_train_full.partner == 'no'].churn.mean()

```
partner_yes = df_train_full[df_train_full.partner == 'yes'].churn.mean()
print('partner == yes:', round(partner_yes, 3))

partner_no = df_train_full[df_train_full.partner == 'no'].churn.mean()
print('partner == no :', round(partner_no, 3))
```

```
partner == yes: 0.205
partner == no : 0.33
```

Risk Ratio

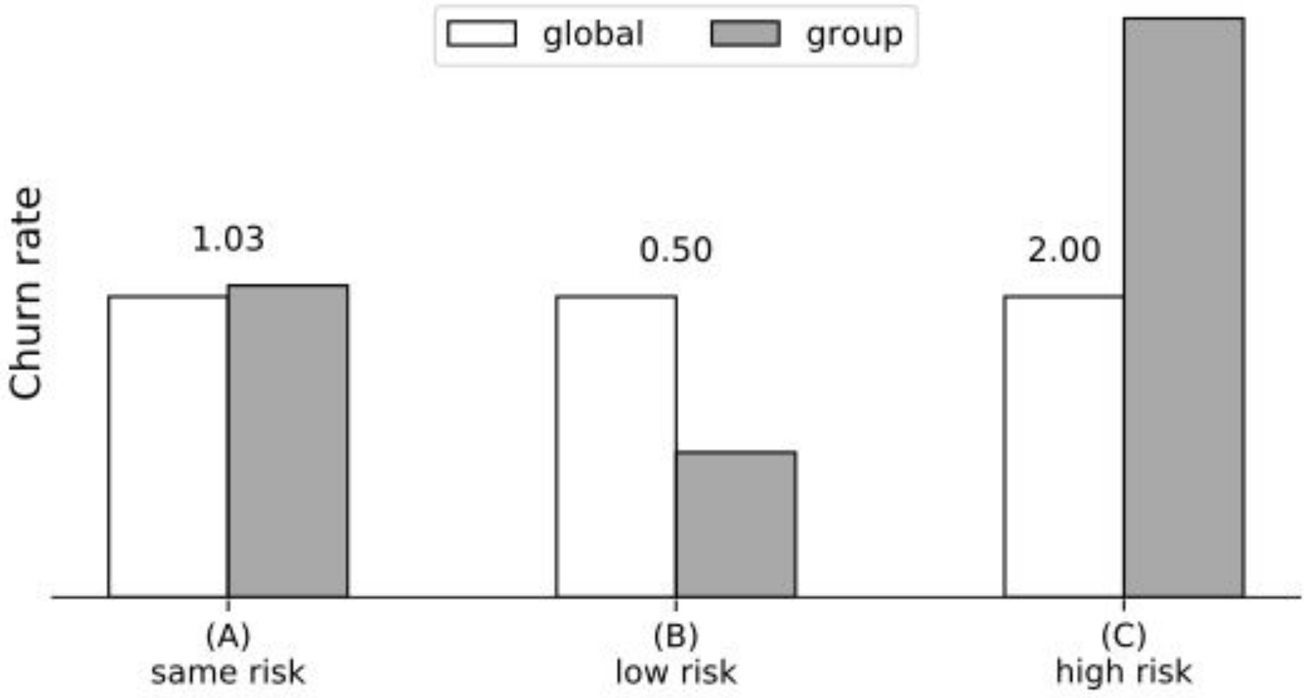
In addition to looking at the difference between the group rate and the global rate, it's interesting to look at the ratio between them. In statistics, the ratio between probabilities in different groups is called risk ratio, where risk refers to the risk of having the effect.

risk = group rate / global rate

For "gender == female", for example, the risk of churning is 1.02:

risk = 27.7% / 27% = 1.02

Risk Ratio


global
group
Churn rate
1.03
0.50
2.00
(A)
same risk
(B)
low risk
(C)
high risk

Context for Risk Ratio

- The term risk originally comes from controlled trials, in which one group of patients is given a treatment (a medicine) and the other group isn't (only a placebo).
- Then we compare how effective the medicine is by calculating the rate of negative outcomes in each group and then calculating the ratio between the rates:

risk = negative outcome rate in group 1 / negative outcome rate in group 2

Risk Table

Variable	Value	Churn rate	Risk
Gender	Female	27.7%	1.02
	Male	26.3%	0.97
Partner	Yes	20.5%	0.75
	No	33%	1.22

SQL versus Pandas (Compute Risk Ratio)

```
SELECT

    gender, AVG(churn),

    AVG(churn) - global_churn,

    AVG(churn) / global_churn

FROM

    data

GROUP BY

    gender
```

```
global_mean = df_train_full.churn.mean()

df_group =
df_train_full.groupby(by='gender').churn.
agg(['mean'])

df_group['diff'] = df_group['mean'] -
global_mean

df_group['risk'] = df_group['mean'] /
global_mean

df_group
```

gender	mean	diff	risk
female	0.276824	0.006856	1.025396
male	0.263214	-0.006755	0.974980

Risk Ratio for All Categorical Variables

```
from IPython.display import display

for col in categorical:

    df_group =
df_train_full.groupby(by=col).churn.agg(['mean'])

    df_group['diff'] = df_group['mean'] - global_mean

    df_group['rate'] = df_group['mean'] / global_mean

    display(df_group)
```

Analysis based on Risk Ratio

gender	mean	diff	risk
female	0.276824	0.006856	1.025396
male	0.263214	-0.006755	0.974980

(A) Churn ratio and risk: gender

seniorcitizen	mean	diff	risk
0	0.242270	-0.027698	0.897403
1	0.413377	0.143409	1.531208

(B) Churn ratio and risk: senior citizen

partner	mean	diff	risk
no	0.329809	0.059841	1.221659
yes	0.205033	-0.064935	0.759472

(C) Churn ratio and risk: partner

phoneservice	mean	diff	risk
no	0.241316	-0.028652	0.893870
yes	0.273049	0.003081	1.011412

(D) Churn ratio and risk: phone service

- **For gender, there is not much difference between females and males.** Both means are approximately the same, and for both groups the risks are close to 1.
- **Senior citizens tend to churn more than nonseniors:** the risk of churning is 1.53 for seniors and 0.89 for nonseniors.
- **People with a partner churn less than people with no partner.** The risks are 0.75 and 1.22, respectively.
- **People who use phone service are not at risk of churning:** the risk is close to 1, and there’s almost no difference with the global churn rate. People who don’t use phone service are even less likely to churn: the risk is below 1, and the difference with the global churn rate is negative.

Churn Analysis

techsupport	mean	diff	risk
no	0.418914	0.148946	1.551717
no_internet_service	0.077805	-0.192163	0.288201
yes	0.159926	-0.110042	0.592390

(A) Churn ratio and risk: tech support

contract	mean	diff	risk
month-to-month	0.431701	0.161733	1.599082
one_year	0.120573	-0.149395	0.446621
two_year	0.028274	-0.241694	0.104730

(B) Churn ratio and risk: contract

- Clients with no tech support tend to churn more than those who do.
- People with monthly contracts cancel the contract a lot more often than others, and people with two-year contacts churn very rarely.

Mutual Information : Categorical

“Customers with month-to-month contracts tend to churn a lot more than customers with other kinds of contracts. This is exactly the kind of relationship we want to find in our data. Without such relationships in data, machine learning models will not work — they will not be able to make predictions. The higher the degree of dependency, the more useful a feature is.”

Mutual Information : Categorical

```
from sklearn.metrics import mutual_info_score

def calculate_mi(series):
    return mutual_info_score(series, df_train_full.churn)

df_mi = df_train_full[categorical].apply(calculate_mi)
df_mi = df_mi.sort_values(ascending=False).to_frame(name='MI')
df_mi
```

Mutual Information : Categorical

	MI
contract	0.098320
onlinesecurity	0.063085
techsupport	0.061032
internetservice	0.055868
onlinebackup	0.046923

(A) The most useful features according to the mutual information score.

	MI
partner	0.009968
seniorcitizen	0.009410
multiplelines	0.000857
phoneservice	0.000229
gender	0.000117

(B) The least useful features according to the mutual information score.

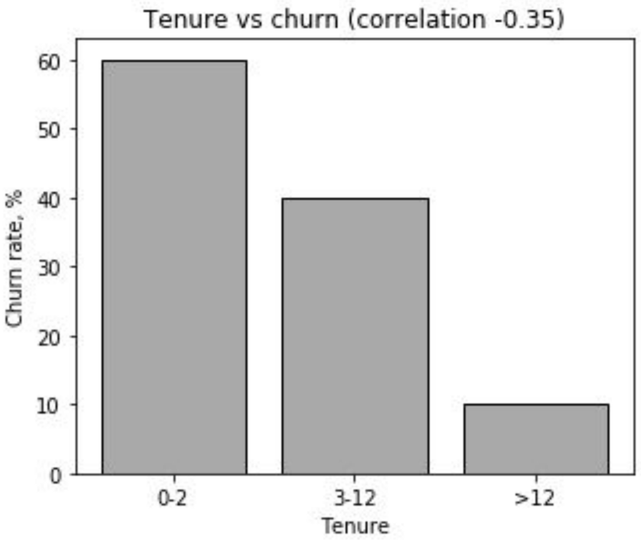
Correlation Coefficient

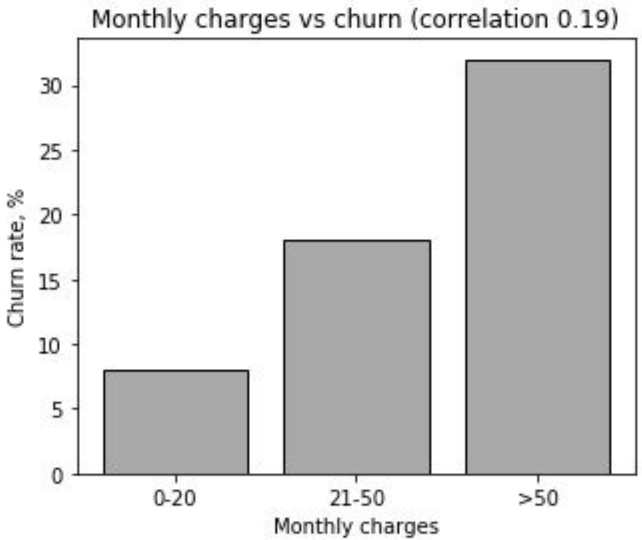
```
df_train_full[numerical].corrwith(df_train_full.churn)
```

Mutual Information shows the degree of dependency of Categorical Variables to the Target Variable.

Correlation does the same with Numeric Variables.

Correlation





	correlation
tenure	-0.351885
monthlycharges	0.196805
totalcharges	-0.196353

Feature Engineering

Transform all categorical variables to numeric forms

One Hot Encoding

One Hot Encoding Concept

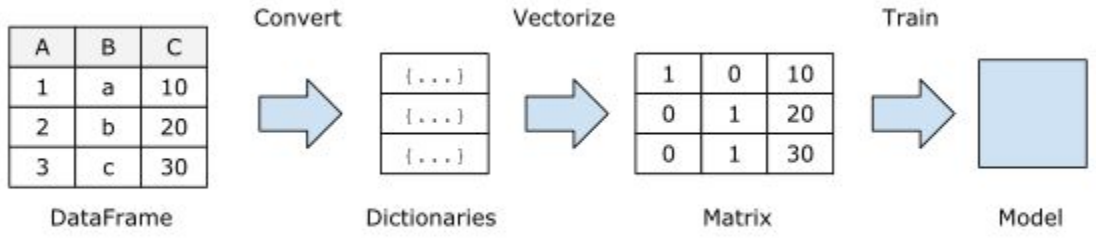
gender	contract
male	monthly
female	yearly

gender		contract		
female	male	monthly	yearly	2-year
0	**1**	**1**	0	0
1	0	0	**1**	0

DictVectorizer

- To use it, we need to convert our dataframe to a list of dictionaries. It’s very simple to do in Pandas. Use the to_dict method with the orient="records" parameter:

```
train_dict = df_train[categorical +
numerical].to_dict(orient='records')
```



```
{'gender': 'male',
 'seniorcitizen': 0,
 'partner': 'yes',
 'dependents': 'yes',
 'phoneservice': 'yes',
 'multiplelines': 'no',
 'internetservice': 'no',
 'onlinesecurity': 'no_internet_service',
 'onlinebackup': 'no_internet_service',
 'deviceprotection': 'no_internet_service',
 'techsupport': 'no_internet_service',
 'streamingtv': 'no_internet_service',
 'streamingmovies': 'no_internet_service',
 'contract': 'two_year',
 'paperlessbilling': 'no',
 'paymentmethod': 'mailed_check',
 'tenure': 12,
 'monthlycharges': 19.7,
 'totalcharges': 258.35}
```

Dictionary Vectorizer

```
from sklearn.feature_extraction import DictVectorizer


dv = DictVectorizer(sparse=False)

dv.fit(train_dict)

X_train = dv.transform(train_dict)
```

Peek at the Vectorized Data

```
X_train[0]
```

```
dv.get_feature_names()
```

Exercise

```
#How would DictVectorizer encode the following list of
#dictionaries:
records = [
    {'total_charges': 10, 'paperless_billing': 'yes'},
    {'total_charges': 30, 'paperless_billing': 'no'},
    {'total_charges': 20, 'paperless_billing': 'no'}
]
```

Machine learning

Predictive Analytics from the clean Telco Dataset

ML for Classification

Logistic Regression

Linear Regression

$$g(x_i) = w_0 + x_i^T w$$

where

xi is the feature vector corresponding to the ith observation,

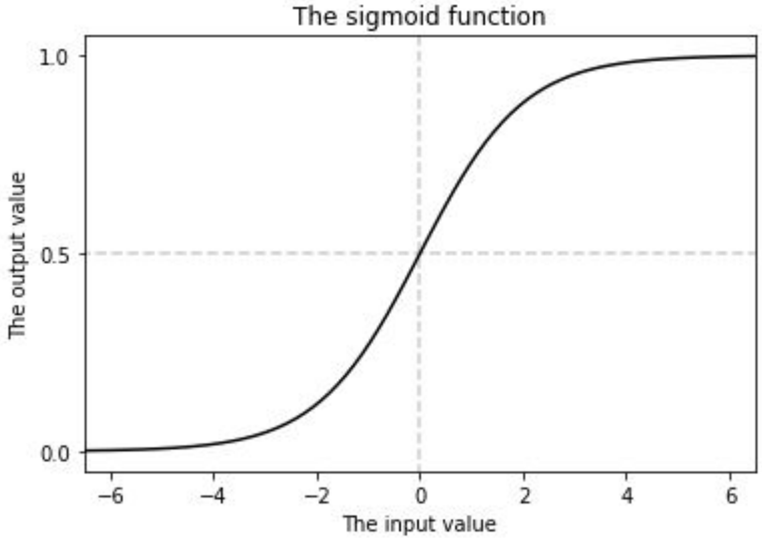
w0 is the bias term,

w is a vector with the weights of the model.

Logistic Regression

$$g(x_i) = \text{sigmoid}(w_0 + x_i^T w)$$

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$



Linear Regression from Scratch in Python

```
def linear_regression(xi):
    result = bias
    for j in range(n):
        result = result + xi[j] * w[j]
    return result
```

Logistic Regression from Scratch using Python

```
import math
def logistic_regression(xi):
    score = bias
    for j in range(n):
        score = score + xi[j] * w[j]
    prob = sigmoid(score)
    return prob
def sigmoid(score):
    return 1 / (1 + math.exp(-score))
```

Exercise

Why do we need the Sigmoid Function for Logistic Regression?

Training the Logistic Regression Model

Training the Model

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(solver='liblinear', random_state=1)

model.fit(X_train, y_train)
```

One Hot Encoding

```
val_dict = df_val[categorical +
numerical].to_dict(orient='records')

X_val = dv.transform(val_dict)

y_pred = model.predict_proba(X_val)
```

Understanding the Predictions

- The predictions of the model: a two-column matrix.
- The first column is the probability that the target=0
- The second column is the probability tha the target=1

```
model.predict_proba(X_val)
```

```
array([[0.76508957, 0.23491043],
       [0.73113584, 0.26886416],
       [0.68054864, 0.31945136],
       ...,
       [0.94274779, 0.05725221],
       [0.38476995, 0.61523005],
       [0.9387273 , 0.0612727 ]])
```

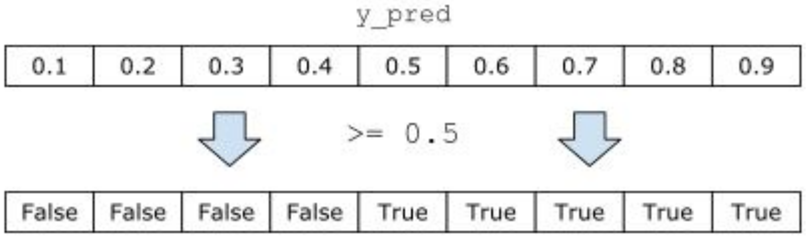
Probability that the observation belongs to the negative class: i.e. customer will not churn

Probability that the observation belongs to the positive class, i.e. customer will churn

But We only need ONE column

```
y_pred = model.predict_proba(X_val)[:, 1]
```

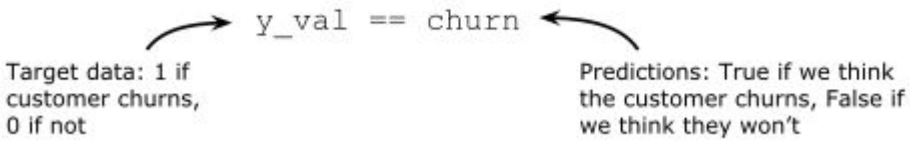
```
y_pred >= 0.5
```



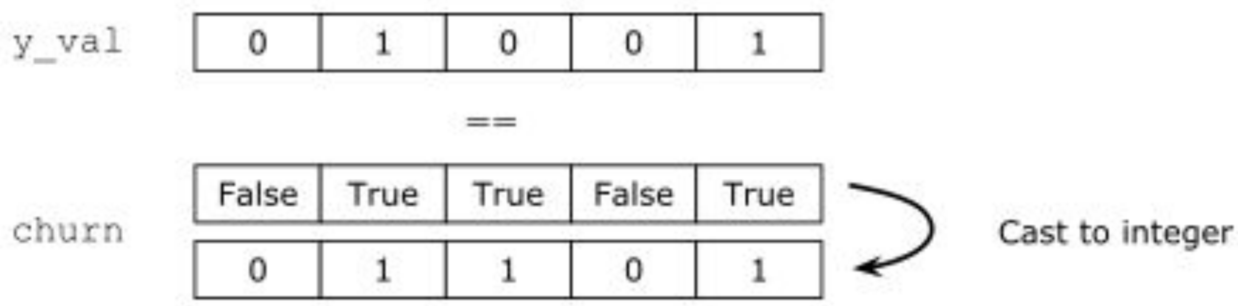
Introducing Accuracy

```
churn = y_pred >= 0.5
```

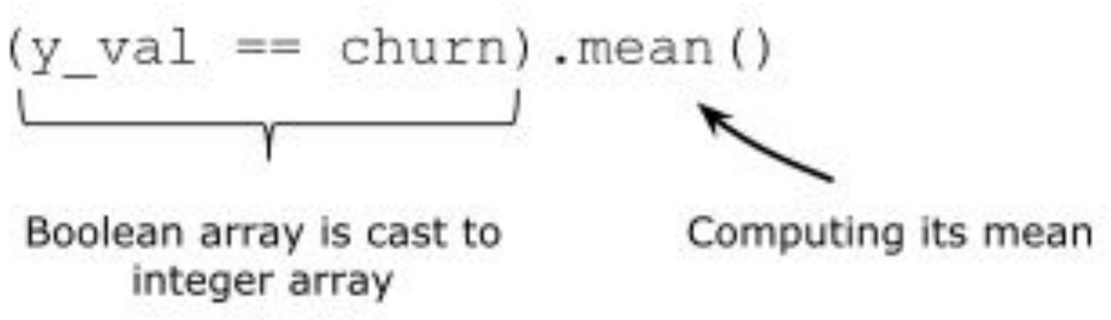
```
(y_val == churn).mean() #Quality Measure called ACCURACY
```



Accuracy


y_val
0
1
0
0
1
==
False
True
True
False
True
churn
0
1
1
0
1
Cast to integer

Accuracy



(y_val == churn).mean()
Boolean array is cast to
integer array
Computing its mean

Model Interpretation

Understand What You've Done : Coefficients

w0 is the bias term.

w = (w1, w2, …, wn) is the weights vector

- We can get the bias term from model.intercept_[0]. When we train our model on all features, the bias term is -0.12
- The rest of the weights are stored in model.coef_[0]

dict(zip(dv.get_feature_names(), model.coef_[0].round(3)))

Weights

```
{'contract=month-to-month': 0.563,
 'contract=one_year': -0.086,
 'contract=two_year': -0.599,
 'dependents=no': -0.03,
 'dependents=yes': -0.092,
 ... # the rest of the weights is omitted
 'tenure': -0.069,
 'totalcharges': 0.0}
```

Prepare a Small Subset to Break Down the Categoricals

```
small_subset = ['contract', 'tenure', 'totalcharges']

train_dict_small =
df_train[small_subset].to_dict(orient='records')

dv_small = DictVectorizer(sparse=False)

dv_small.fit(train_dict_small)

X_small_train = dv_small.transform(train_dict_small)

dv_small.get_feature_names()
```

```
['contract=month-to-month',
 'contract=one_year',
 'contract=two_year',
 'tenure',
 'totalcharges']
```

Train the Small Subset

```
model_small = LogisticRegression(solver='liblinear',
random_state=1)

model_small.fit(X_small_train, y_train)

model_small.intercept_[0] #Check the bias

dict(zip(dv_small.get_feature_names(),
model_small.coef_[0].round(3)))  #Check the other weights
```

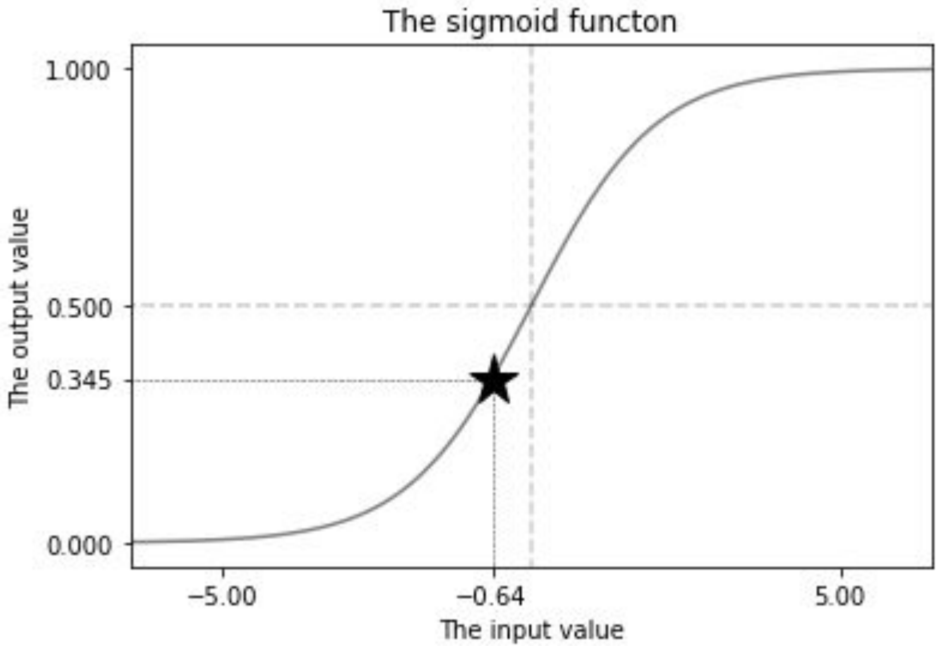
```
{'contract=month-to-month': 0.91,
 'contract=one_year': -0.144,
 'contract=two_year': -1.404,
 'tenure': -0.097,
 'totalcharges': 0.000}
```

The Big Picture with the Categoricals

bias	contract			tenure	charges
	month	year	2-year		
w_0	w_1	w_2	w_3	w_4	w_5
-0.639	0.91	-0.144	-1.404	-0.097	0.0

The Sigmoid Function

The bias term -0.639 on the sigmoid curve. The resulting probability is less than 0.5 so the average customer is more likely not to churn.



Understanding The Importance of Categories

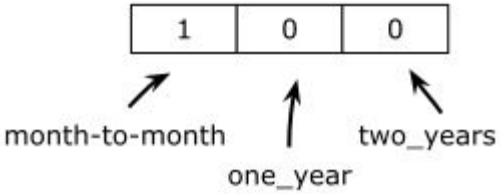
```
dict(zip(dv_small.get_feature_names(),
model_small.coef_[0].round(3)))
```

```
'contract=month-to-month': 0.91,

 'contract=one_year': -0.144,

 'contract=two_year': -1.404.
```

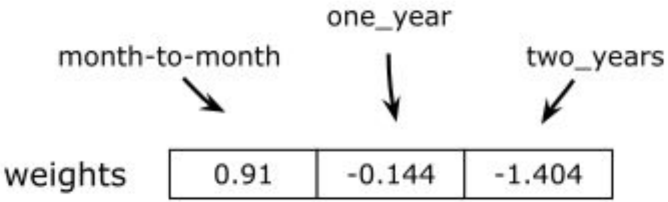
```
{'contract=month-to-month': 0.91,
 'contract=one_year': -0.144,
 'contract=two_year': -1.404,
 'tenure': -0.097,
 'totalcharges': 0.000}
```

Build your Intuition

The one-hot encoded representation for a customer with a month to month contract

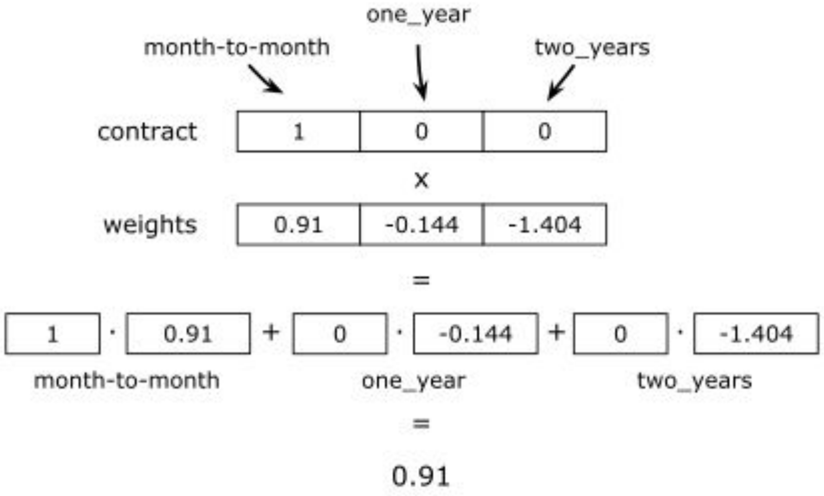


The weights of the month to month, 1 year, 2 year contract features



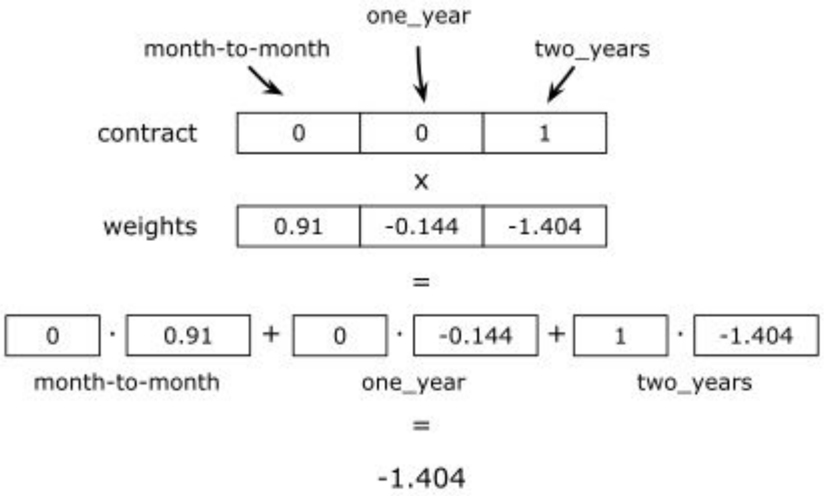
Making a Prediction with One Hot Encoded Categorical Data

- The dot product between the one-hot encoding representation of the contract variable and the corresponding weights.
- The result is 0.91 which is the weight of the hot feature.



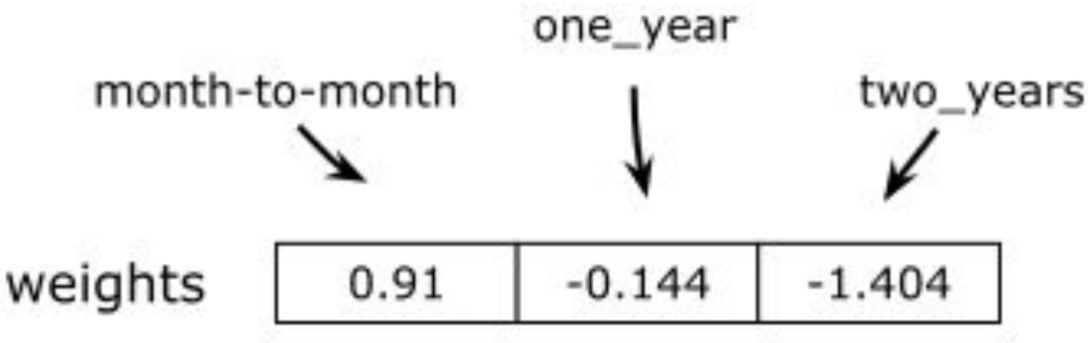
Same Thing but a 2 year contract client

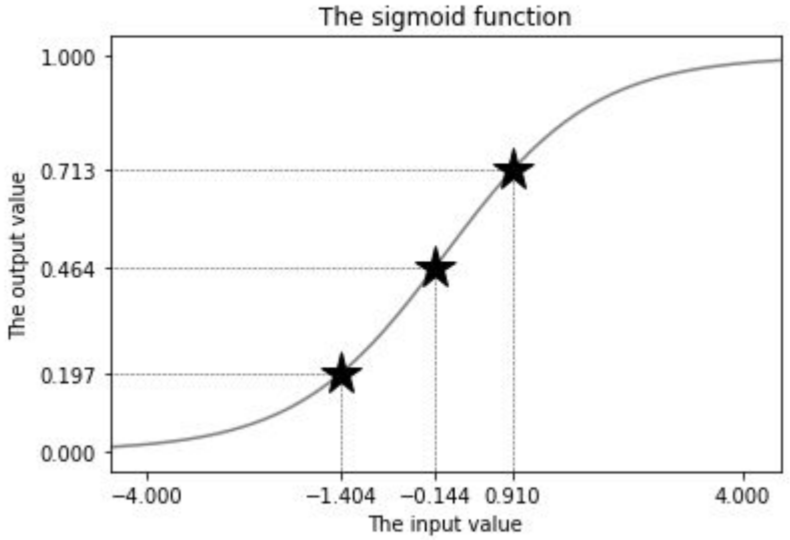
Only the HOT feature is ever factored into the model.



Does the sign (+/-) have meaning?

The sign of the weight matters. If it is positive - churn. Negative - loyal customer.





Let's Analyze the Numerical Features

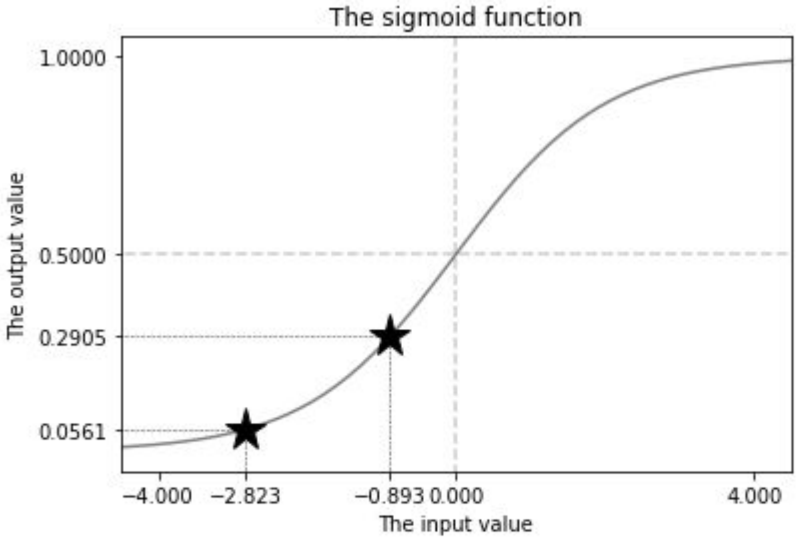
The score model calculate for a customer with a m2m contract, 12 months of tenure and total charges of $1,000

$$\underbrace{-0.639}_{\text{bias}} \underbrace{+ 0.91}_{\substack{\text{monthly}\\\text{contract}}} \underbrace{- 12 \cdot 0.097}_{\substack{\text{12 months}\\\text{of}\\\text{tenure}}} \underbrace{+ 0 \cdot 1000}_{\substack{\text{total charges}\\\text{don't}\\\text{matter}}} = \underbrace{-0.893}_{\substack{\text{negative, so}\\\text{low likelihood of}\\\text{churn}}}$$

The score model calculate for a customer with a yearly contract, 24 months of tenure and total charges of $2,000

$$-0.639 + 0.144 - 24 \cdot 0.097 + 0 \cdot 2000 = \ -2.823$$

bias	yearly contract	24 months of tenure	total charges don't matter	negative, very low likelihood of churn



Using the Churn Model

```
customer = {
    'customerid': '8879-zkjof',
    'gender': 'female',
    'seniorcitizen': 0,
    'partner': 'no',
    'dependents': 'no',
    'tenure': 41,
    'phoneservice': 'yes',
    'multiplelines': 'no',
    'internetservice': 'dsl',
    'onlinesecurity': 'yes',
    'onlinebackup': 'no',
```

```
#Continuation
'deviceprotection': 'yes',
    'techsupport': 'yes',
    'streamingtv': 'yes',
    'streamingmovies': 'yes',
    'contract': 'one_year',
    'paperlessbilling': 'yes',
    'paymentmethod': 'bank_transfer_(a
    'monthlycharges': 79.85,
    'totalcharges': 3320.75,
}
```

Vectorized Input

```
X_test = dv.transform([customer])

#Output

[[   0.  ,    1.  ,    0.  ,    1.  ,    0.  ,    0.  ,    0.  ,
     1.  ,    1.  ,    0.  ,    1.  ,    0.  ,    0.  ,   79.85,
     1.  ,    0.  ,    0.  ,    1.  ,    0.  ,    0.  ,    0.  ,
     0.  ,    1.  ,    0.  ,    1.  ,    1.  ,    0.  ,    1.  ,
     0.  ,    0.  ,    0.  ,    0.  ,    1.  ,    0.  ,    0.  ,
     0.  ,    1.  ,    0.  ,    0.  ,    1.  ,    0.  ,    0.  ,
     1.  ,   41.  , 3320.75]]
```

Put the Matrix into the Trained Model

```
model.predict_proba(X_test)
```

```
#output
```

```
[[0.93, 0.07]]
```

```
X_test = dv.transform([customer])
model.predict_proba(X_test)[0, 1]
```

```
0.07332577315357781
```

All we need from the matrix is the number at the first row and second column: the probability of churning for this customer. To select this number from the array, we use the brackets operator:

`model.predict_proba(X_test)[0, 1]`

Take a look at another customer

```
customer = {
    'gender': 'female',
    'seniorcitizen': 1,
    'partner': 'no',
    'dependents': 'no',
    'phoneservice': 'yes',
    'multiplelines': 'yes',
    'internetservice': 'fiber_optic',
    'onlinesecurity': 'no',
    'onlinebackup': 'no',
    'deviceprotection': 'no',
    'techsupport': 'no',
    'streamingtv': 'yes',
    'streamingmovies': 'no',
    'contract': 'month-to-month',
    'paperlessbilling': 'yes',
    'paymentmethod': 'electronic_check',
    'tenure': 1,
    'monthlycharges': 85.7,
    'totalcharges': 85.7
}
```

Let's Make a Prediction

```
X_test = dv.transform([customer])

model.predict_proba(X_test)[0, 1]
```

```
In [62]: X_test = dv.transform([customer])
         model.predict_proba(X_test)[0, 1]

Out[62]: 0.8321638622459152
```

Project

- Classification models are often used for marketing purposes, and one of the problems it solves is lead scoring.
- A lead is a potential customer who may convert (became an actual customer) or not.
- In this case, the conversion is the target we want to predict.
- You can take a dataset from **https://www.kaggle.com/ashydv/leads-dataset** and build a model for that.
- You may notice that the lead scoring problem is very similar to churn prediction, but in one case we want to get a new client to sign a contract with us, and in another case we want a client not to cancel the contract.

Project 2

- Another popular application of classification is default prediction, which is estimating the risk of a customer's not returning a loan.
- In this case, the variable we want to predict is default, and it also has two outcomes: whether the customer managed to pay back the loan in time (good customer) or not (default).
- There are many datasets online that you can use for training a model, such as https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients (or, the same one available via kaggle: https://www.kaggle.com/pratjain/credit-card-default).

Summary

Classification