A Guide to the Naive Bayes Algorithm

Surabhi S — Published On January 16, 2021 and Last Modified On June 20th, 2022

Algorithm Beginner Classification Machine Learning Maths Python Structured Data

This article was published as a part of the Data Science Blogathon.

Introduction to Naive Bayes algorithm

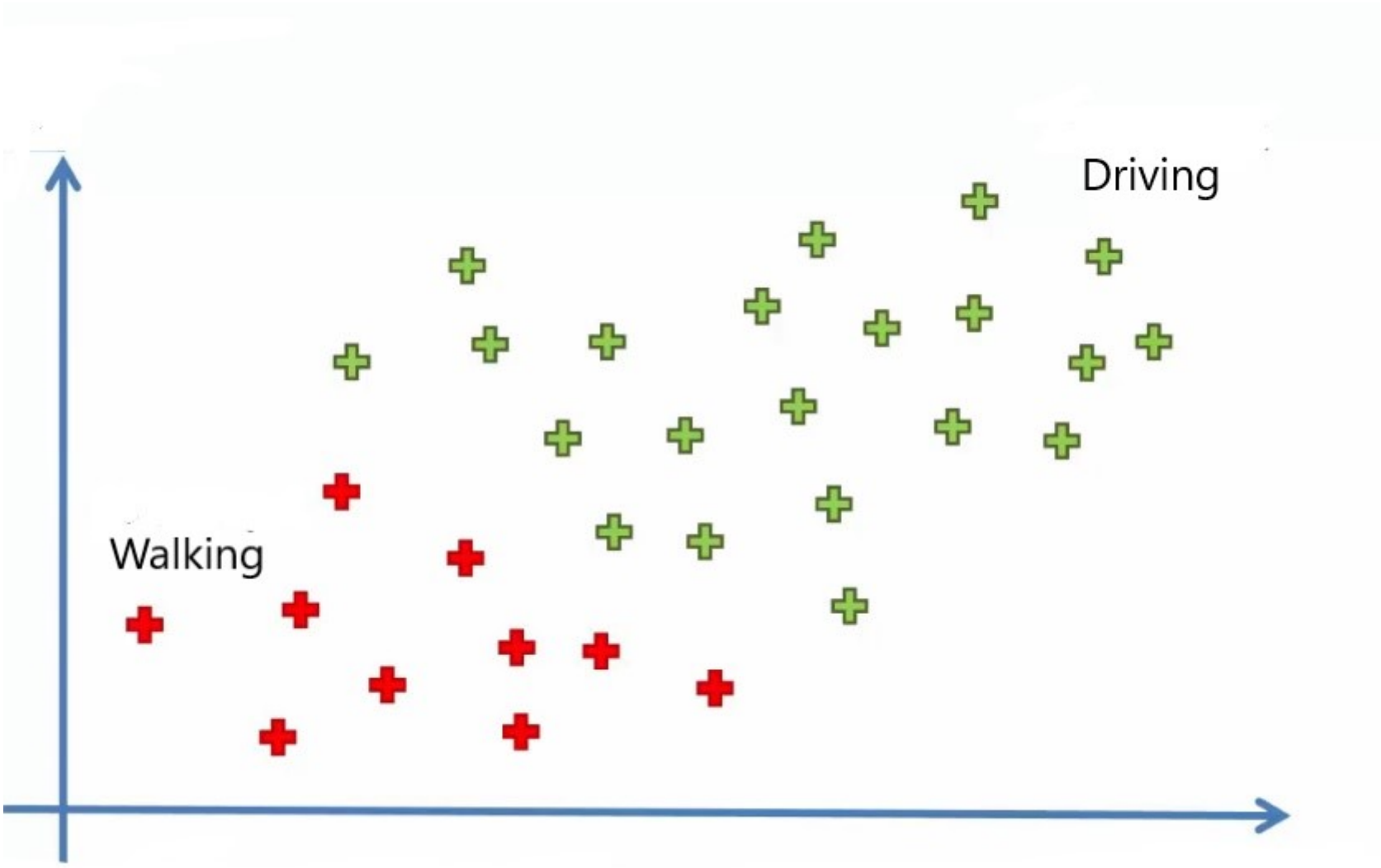
Naive Bayes is a classification algorithm that works based on the Bayes theorem. Before explaining about Naive Bayes, first, we should discuss Bayes Theorem. Bayes theorem is used to find the probability of a hypothesis with given evidence.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

In this, using Bayes theorem we can find the probability of A, given that B occurred. A is the hypothesis and B is the evidence.

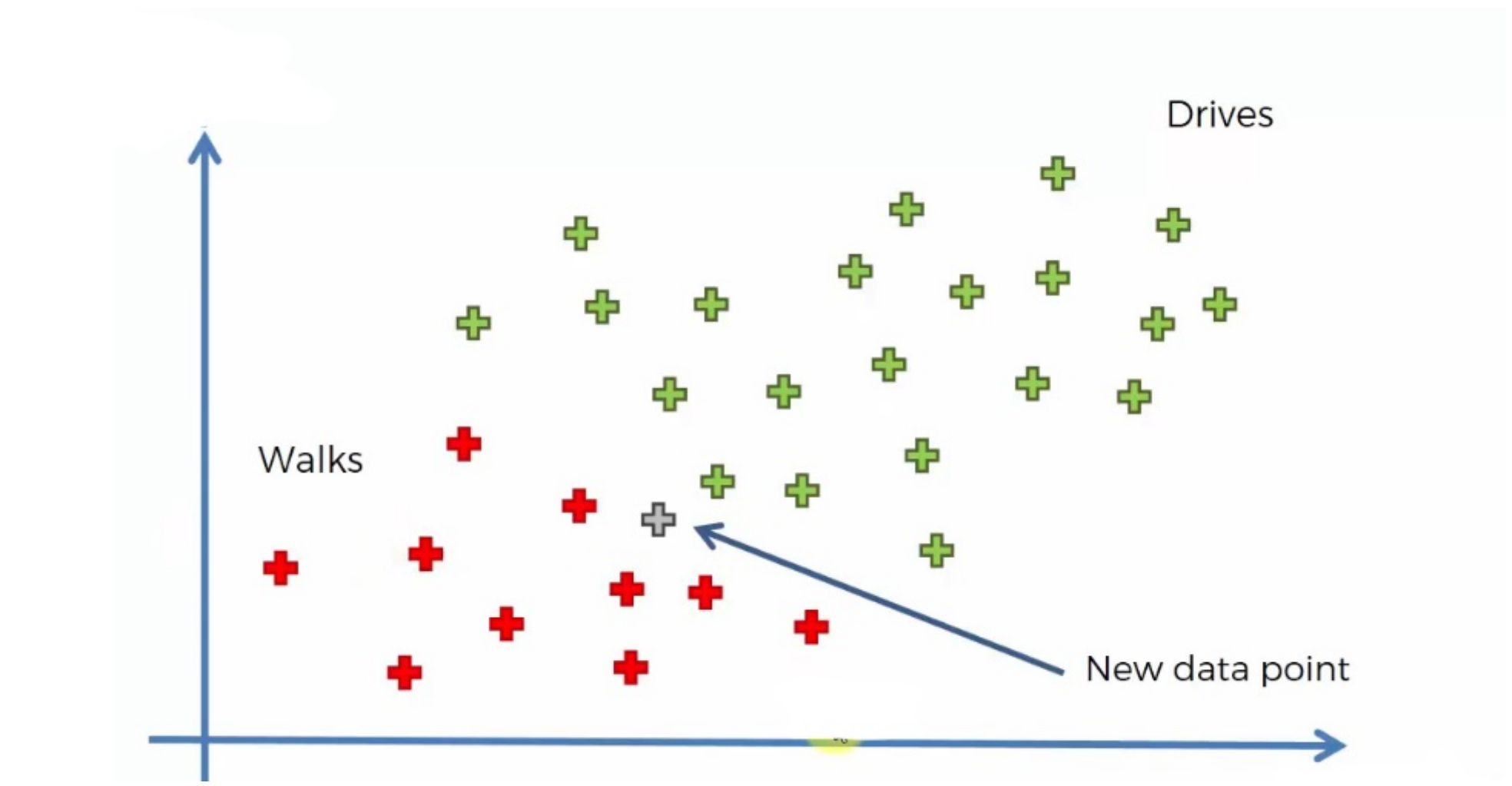
P(B|A) is the probability of B given that A is True.

P(A) and P(B) is the independent probabilities of A and B.

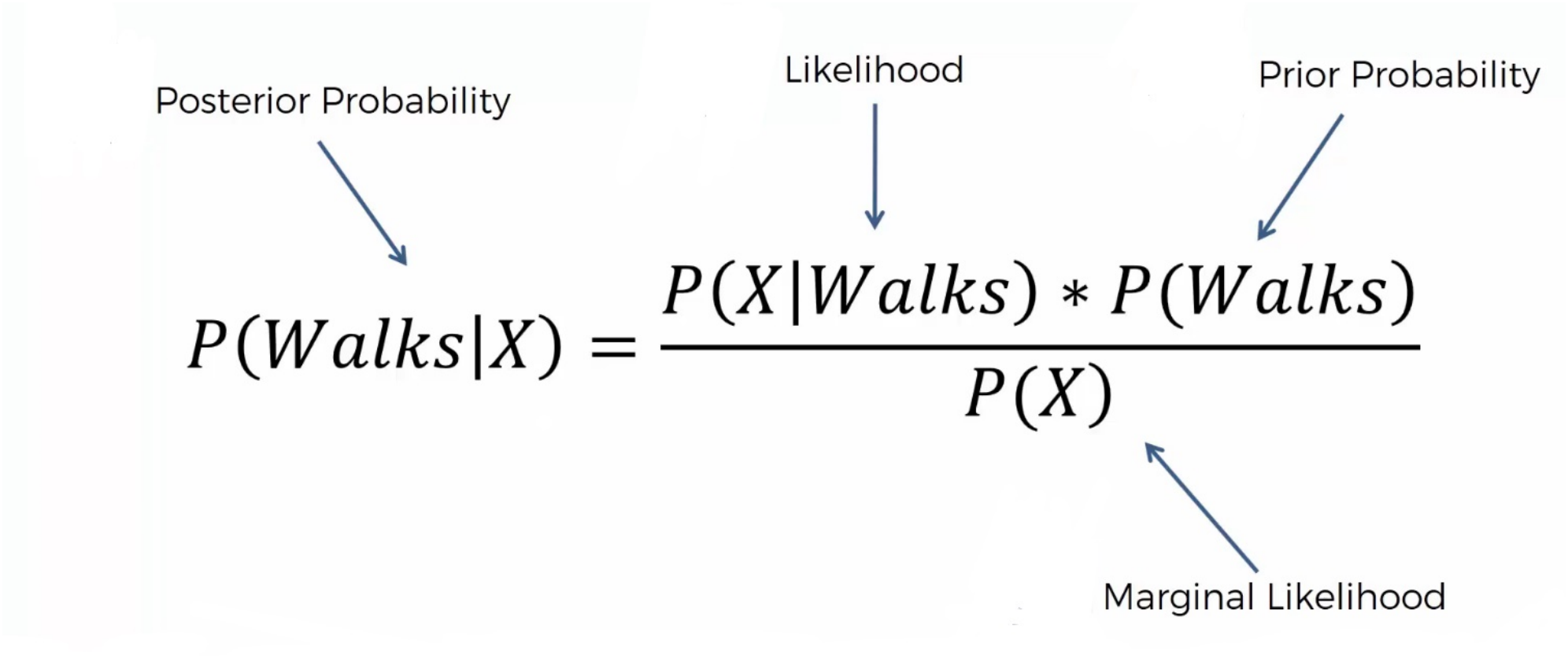


The concept behind the algorithm

Let’s understand the concept of the Naive Bayes Theorem through an example. We are taking a dataset of employees in a company, our aim is to create a model to find whether a person is going to the office by driving or walking using salary and age of the person.



In the above, we can see 30 data points in which red points belong to those who are walking and green belongs to those who are driving. Now let's add a new data point into it. Our aim is to find the category that the new point belongs to.



Note that we are taken age on the X-axis and Salary on the Y-axis. We are using the Naive Bayes algorithm to find the category of the new data point. For this, we have to find the posterior probability of walking and driving for this data point. After comparing, the point belongs to the category having a higher probability.

The posterior probability of walking for the new data point is :

also for the driving is :

Steps involved in Naive Bayes algorithm

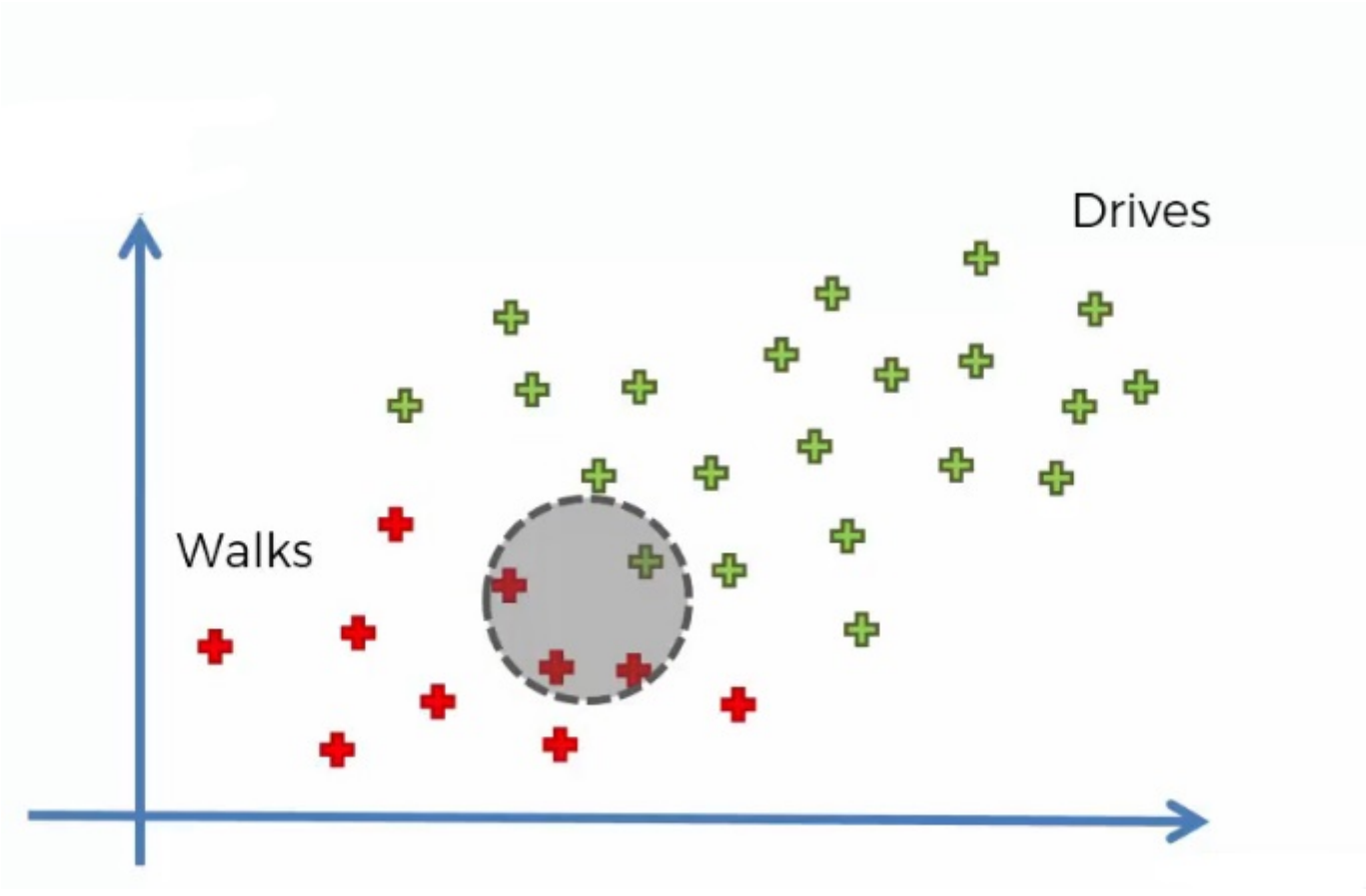
Step 1: We have to find all the probabilities required for the Bayes theorem for the calculation of posterior probability

P(Walks) is simply the probability of those who walk among all

$$P(Walks) = \frac{Number\ of\ Walkers}{Total\ Observations}$$

$$P(Walks) = \frac{10}{30}$$

In order to find the marginal likelihood, P(X), we have to consider a circle around the new data point of any radii including some red and green points.



$$P(X) = \frac{Number\ of\ Similar\ Observations}{Total\ Observations}$$

$$P(X) = \frac{4}{30}$$

P(X|Walks) can be found by :

$$P(X|Walks) = \frac{Number\ of\ Similar\ Observations\ Among\ those\ who\ Walk}{Total\ number\ of\ Walkers}$$

$$P(X|Walks) = \frac{3}{10}$$

$$P(Walks|X) = \frac{\frac{3}{10} * \frac{10}{30}}{\frac{4}{30}} = 0.75$$

Step 2: Similarly we can find the posterior probability of Driving, and it is 0.25

Step 3: Compare both posterior probabilities. When comparing the posterior probability, we can find that P(walks|X) has greater values and the new point belongs to the walking category.

source: Unsplash

Implementation of Naive Bayes

Now let's implement Naive Bayes using python

We are using the Social network ad dataset. The dataset contains the details of users in a social networking site to find whether a user buys a product by clicking the ad on the site based on their salary, age, and gender.

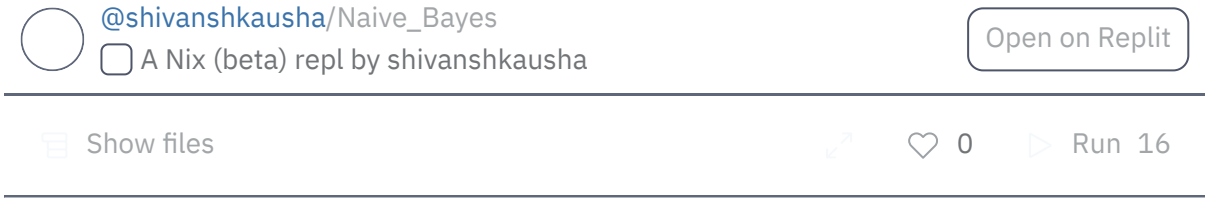
Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0

Let’s start the programming by importing essential libraries required

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn
```

Importing of dataset

Python Code:



Since our dataset containing character variables we have to encode it using LabelEncoder

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:,0] = le.fit_transform(X[:,0])
```

Train test Split

We are performing a train test split on our dataset. We are providing the test size as 0.20, that means our training sample contains 320 training set and test sample contains 80 test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

Feature scaling

Next, we are doing **feature scaling** to the training and test set of independent variables

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Training the Naive Bayes model on the training set

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

Let's predict the test results

```
y_pred  =  classifier.predict(X_test)
```

Predicted and actual value –

```
 y_pred
```

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0

```
y_test
```

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0

For the first 8 values, both are the same. We can evaluate our matrix using the confusion matrix and accuracy score by comparing the predicted and actual test values

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pred)
ac = accuracy_score(y_test,y_pred)
```

confusion matrix –
ac – **0.9125**

	0	1
0	55	3
1	4	18

Accuracy is good. Note that, you can achieve better results for this problem using different algorithms.

Conclusion

Now that we have dealt with the Naive Bayes algorithm, we have covered most concepts of it in machine learning. Do not forget to practice algorithms.

Full Code

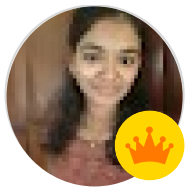
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
# Training the Naive Bayes model on the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
ac = accuracy_score(y_test,y_pred)
cm = confusion_matrix(y_test, y_pred)
```

The media shown in this article are not owned by Analytics Vidhya and is used at the Author's discretion.

blogathon Naive Bayes

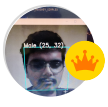
About the Author



Our Top Authors







view more

Download

Analytics Vidhya App for the Latest blog/Article

Previous Post

Python Code Performance Measurement – Measure the right metric to optimize better!

Next Post

AIaaS – Out of the box pre-built Solutions

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Email*

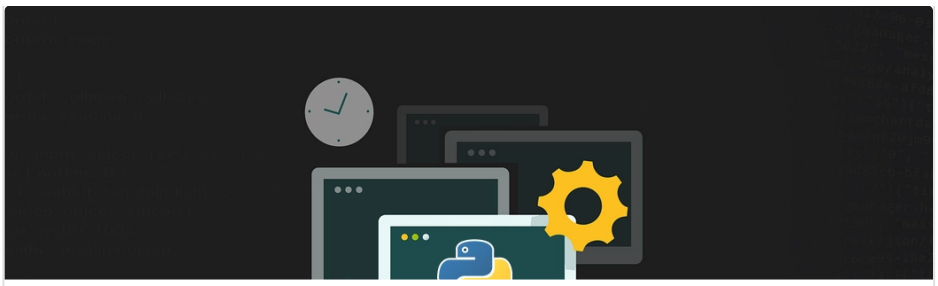
Website

Notify me of follow-up comments by email.

Notify me of new posts by email.

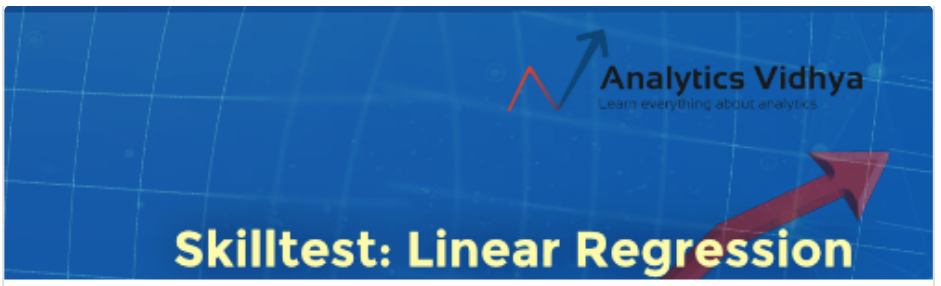
Submit

Top Resources



Python Tutorial: Working with CSV file for Data Science

Harika Bonthu - AUG 21, 2021

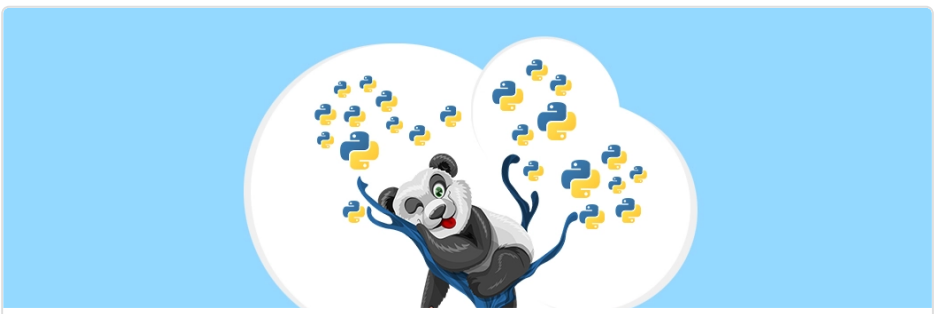


30 Questions to test a data scientist on Linear Regression..

1201904 - JUL 03, 2017

30 Questions to test your understanding of Logistic Regression

1201904 - AUG 03, 2017



Joins in Pandas: Master the Different Types of Joins in..

Abhishek Sharma - FEB 27, 2020