Naïve Bayes Classifier
ML ALGORITHM
Python

DATA SCIENCE
INSTITUTE

Naive Bayes Classifier

- A **naive Bayes classifier** is a simple probabilistic classifier based on Bayes' theorem.

- It can be used as an alternative method to Logistic Regression

- It is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.

DATA SCIENCE INSTITUTE

Bayes Theorem

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

Where

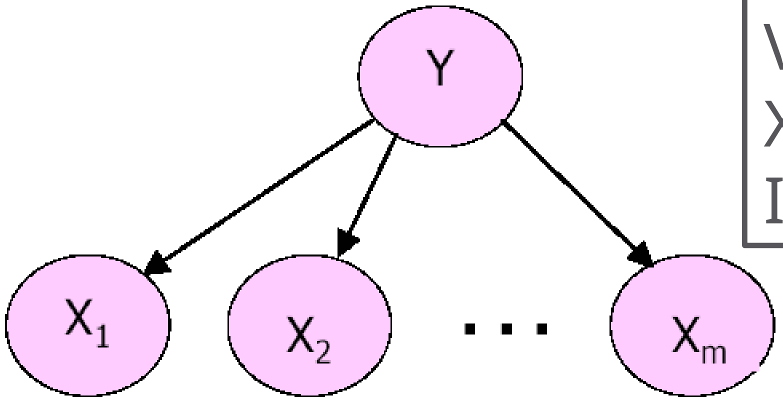
P(A) is the prior probability or marginal probability of A.

P(A|B) is the conditional probability of A, given B.

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

P(B) is the prior or marginal probability of B.

Naïve Bayes Framework



Y : Categorical Dependent Variable
X_i : Categorical/Continuous Independent Variable

Objective: To estimate Y given the values of X_i's or $P(Y | X_1\ X_2\ \ldots X_m)$ using the Naïve Bayes Classifier.

Assumption: All X_i's are conditionally independent of each other.

DATA SCIENCE INSTITUTE

Example

Consider a simple example where dependent variable Y is binary variable and there are 2 independent

variables X1 and X2.

We classify Y = 1 as potential buyer of a certain product

Y = 0 otherwise

Let X1 denote age of the individual

X1 = 0 for age group 25-30 yrs.

= 1 for age group 31-40 yrs.

Let X2 denote gender

X2 = 0 if Gender=female

= 1 if Gender=male

Classification Rule

For the given values of age(X1)and gender(X2), we want to classify the customer as potential buyer or not.

Using Naïve Bayes Classifier we estimate following 2 conditional probabilities:

$$P(Y=0/X1=a1, X2=a2) \text{ and } P(Y=1/X1=a1,X2=a2);$$

here a1 and a2 are values of X1 and X2 for a particular customer.

We classify $Y = 0$ if $P(Y=0/X1=a1, X2=a2) > 0.5$

OR $Y = 1$ if $P(Y=1/X1=a1,X2=a2) > 0.5$

Expected Output

Once the classification rule is applied the output can be shown as follows:

Case#	X1	X2	P(Y=1/X1,X2)	P(Y=0/X1,X2)	Y classified as
1	1	0	0.29	0.71	0
2	1	1	0.65	0.35	1
240	0	0	0.51	0.49	1

DATA SCIENCE INSTITUTE

Case Study – Modeling Loan Defaults

Background

- A bank possesses demographic and transactional data of its loan customers. If the bank has a model to predict defaulters it can help in loan disbursal decision making.

Objective

- To predict whether the customer applying for the loan will be a defaulter or not.

Available Information

- Sample size is 700
- Independent Variables: Age group, Years at current address, Years at current employer, Debt to Income Ratio, Credit Card Debts, Other Debts. The information on predictors was collected at the time of loan application process.
- Dependent Variable: Defaulter (=1 if defaulter ,0 otherwise). The status is observed after loan is disbursed.

Data Snapshot

Independent Variables　　　　Dependent Variable

SN	AGE	EMPLOY	ADDRESS	DEBTINC	CREDDEBT	OTHDEBT	DEFAULTE
1	3	17	12	9.3	11.36	5.01	1
2	1	10	6	17.3	1.36	4	0
3	2	15	14	5.5	0.86	2.17	0
4	3	15	14	2.9	2.66	0.82	0

Column	Description	Type	Measurement	Possible Values
SN	Serial Number	numeric	-	-
AGE	Age Groups	Integer	1(<28 years), 2(28-40 years), 3(>40 years)	3
EMPLOY	Number of years customer working at current employer	Integer	-	Positive value
ADDRESS	Number of years customer staying at current address	Integer	-	Positive value
DEBTINC	Debt to Income Ratio	Continuous	-	Positive value
CREDDEBT	Credit to Debit Ratio	Continuous	-	Positive value

Naive Bayes Method in Python

```
# Importing required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB, MultinomialNB

from sklearn.metrics import confusion_matrix, f1_score,
precision_score, recall_score, accuracy_score,
roc_curve, roc_auc_score
```

- Naive Bayes methods differ based on the type of predictors- continuous or categorical
- Python's sklearn has various methods available and the two methods explored hereon are Gaussian Naive Bayes and Multinomial Naive Bayes.

Naive Bayes Method in Python For Continuous Predictors

```
# Importing and Readying the Data for Modeling
bankloan = pd.read_csv("BANK LOAN.csv")
bankloan1 = bankloan.drop(['SN','AGE'], axis = 1)
bankloan1.head()
```

- **drop()** is used to remove unwanted variables. AGE is removed because it is a categorical variable.
- **axis = 1** drops columns.

```
# Output
```

	EMPLOY	ADDRESS	DEBTINC	CREDDEBT	OTHDEBT	DEFAULTER
0	17	12	9.3	11.36	5.01	1
1	10	6	17.3	1.36	4.00	0
2	15	14	5.5	0.86	2.17	0
3	15	14	2.9	2.66	0.82	0
4	2	0	17.3	1.79	3.06	1

Naive Bayes Method in Python For Continuous Predictors

```
# Creating Train and Test Data Sets
X = bankloan1.loc[:,bankloan1.columns != 'DEFAULTER']
y = bankloan1.loc[:, 'DEFAULTER']

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.30,
                                                    random_state = 999)
```

- **train_test_split()** from sklearn.model_selection is used to split dataset into random train and test sets.
- **test_size** represents the proportion of dataset to be included in the test set.
- **random_state** sets the seed for the random number generator.

DATA SCIENCE INSTITUTE

Naive Bayes Method in Python For Continuous Predictors

```
# Model Fitting
NBmodel = GaussianNB()

NBmodel.fit(X_train, y_train)
```

- **GaussianNB()** fits a Gaussian Naive Bayes algorithm for classification.
- This model is suitable for continuous predictors and assumes the likelihood of predictors to be normal.

```
# Predicted Probabilities
predprob_test = NBmodel.predict_proba(X_test)
predprob_test
```

- **predict_proba()** returns predicted probabilities for the test data.

```
# Output
array([[0.96499091, 0.03500909],
       [0.86941828, 0.13058172],
       [0.90585744, 0.09414256],
       [0.97398393, 0.02601607],
       [0.99549445, 0.00450555],
       [0.52978724, 0.47021276],
```

Naive Bayes Method in Python For Continuous Predictors

```
# Custom Cutoff Value for Prediction Labels
cutoff = 0.3
pred_test = np.where(predprob_test[:,1] > cutoff, 1, 0)
pred_test
```

- The output is an array of binary labels.

```
# Output
```

```
array([0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0])
```

Naive Bayes Method in Python
For Continuous Predictors

```
# Confusion Matrix
confusion_matrix(y_test, pred_test, labels=[0, 1])

array([[135, 22],
       [ 26, 27]])

accuracy_score(y_test, pred_test)
0.7714285714285715

precision_score(y_test, pred_test)
0.5510204081632653

recall_score(y_test, pred_test)
0.5094339622641509
```

- **accuracy_score() =** number of correct predictions out of total predictions
- **precision_score()** = true positives / (true positives + false positives)
- **recall_score()** also known as 'Sensitivity' = true positives / (true positives + false negatives)

```
# Area Under ROC Curve
auc = roc_auc_score(y_test, predprob_test[:,1])
print('AUC: %.3f' % auc)
AUC: 0.816
```

- **roc_auc_score** computes Area Under the ROC curve.

Note : Output will be slightly different as observations are randomly assigned to train-test data.

Naive Bayes Method in Python For Continuous Predictors

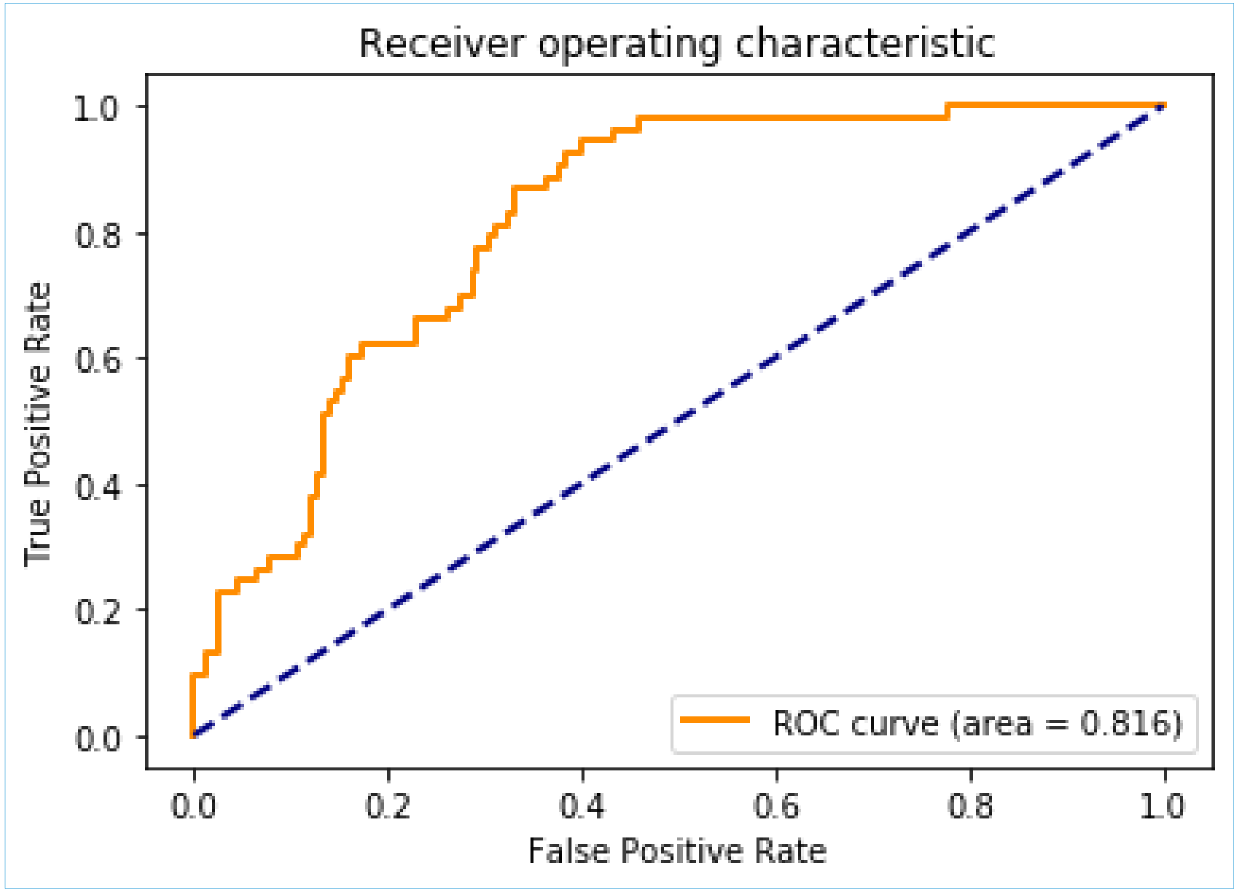
```
# ROC Curve
NBfpr, NBtpr, thresholds = roc_curve(y_test, predprob_test[:,1])

# plot the roc curve for the model
plt.figure()
lw = 2
plt.plot(NBfpr, NBtpr, color='darkor
(area = %0.3f)' % auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.axis('tight')
plt.xlabel('False Positive Rate');plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

- **roc_curve** is used to Compute Receiver operating characteristic.

Naive Bayes Method in Python For Continuous Predictors

```
# Output:
```



Case Study – Employee Churn Model

Background

- A company has comprehensive database of its past and present workforce, with information on their demographics, education, experience and hiring background as well as their work profile. The management wishes to see if this data can be used for predictive analysis, to control attrition levels.

Objective

- To develop an Employee Churn model via Naive Bayes

Available Information

- Sample size is 83
- Gender, Experience Level (<3, 3-5 and >5 years), Function (Marketing, Finance, Client Servicing (CS)) and Source (Internal or External) are independent variables
- Status is the dependent variable (=1 if employee left within 18 months from joining date)

DATA SCIENCE INSTITUTE

Data Snapshot

EMPLOYEE CHURN DATA

Dependent Variable

Independent Variables

sn	status	function	exp	gender	source
1	1	CS	<3	M	external

Columns	Description	Type	Measurement	Possible values
sn	Serial Number	Integer	-	-
status	= 1 If the Employee Left Within 18 Months of Joining = 0 Otherwise	Integer	1,0	2
function	Employee Job Profile	Character	CS, FINANCE, MARKETING	3
exp	Experience in Years	Character	<3,3-5,>5	3
gender	Gender of the Employee	Character	M,F	2
source	Whether the Employee was Appointed via Internal or External Links	Character	external, internal	2

DATA SCIENCE INSTITUTE

Naive Bayes Method in Python
For Categorical Predictors

```
# Importing and Readying the Data for Modeling, Model Fitting
empdata = pd.read_csv("EMPLOYEE CHURN DATA.csv")
empdata1 = empdata.loc[:, empdata.columns != 'sn']
empdata1.head()
```

- **loc()** is used to create a subset of the data frame using column name. Removing column with serial numbers.

```
# Output
```

	status	function	exp	gender	source
0	1	CS	<3	M	external
1	1	CS	<3	M	external
2	1	CS	>=3 and <=5	M	internal
3	1	CS	>=3 and <=5	F	internal
4	1	CS	<3	M	internal

Naive Bayes Method in Python
For Categorical Predictors

```
# Creating Dummy Variables
```

```
empdata2 = pd.get_dummies(empdata1)
empdata2.head()
```

- **pd.get_dummies()** converts categorical variables into dummy variables. This step is crucial because the naive Bayes function used for categorical variables requires this format.

```
# Output
```

```
   status  function_CS  ...  source_external  source_internal
0       1            1  ...                1                0
1       1            1  ...                1                0
2       1            1  ...                0                1
3       1            1  ...                0                1
4       1            1  ...                0                1
```

Naive Bayes Method in Python For Categorical Predictors

```
# Creating Data Partitions
X_emp = empdata2.loc[:,empdata2.columns != 'status']
y_emp = empdata2.loc[:, 'status']
```

```
# Model Fitting
MNBmodel = MultinomialNB(alpha = 0)

MNBmodel.fit(X_emp, y_emp)
```

- **MultinomialNB()** fits a Multinomial Naive Bayes algorithm for classification. This model is suitable for categorical predictors.
- alpha = 0 ensures the model doesn't apply any smoothing on the data.

Naive Bayes Method in Python For Categorical Predictors

```
# Predicted Probabilities
```

```
predprob_MNB = MNBmodel.predict_proba(X_emp)
predprob_MNB
```

```
# Output
```

```
array([[0.06419224, 0.93580776],
       [0.06419224, 0.93580776],
       [0.49966736, 0.50033264],
       [0.5358654 , 0.4641346 ],
       [0.1377729 , 0.8622271 ],
       [0.5625865 , 0.4374135 ],
       [0.59789573, 0.40210427],
       [0.07347548, 0.92652452],
```

```
# Custom Cutoff Value for Prediction Labels
```

```
cutoff = 0.3
pred_test = np.where(predprob_MNB[:,1] > cutoff, 1, 0)
```

Naive Bayes Method in Python For Categorical Predictors

```
# Confusion Matrix
confusion_matrix(y_emp, pred_test, labels=[0, 1])

array([[37, 13],
       [ 3, 30]])

accuracy_score(y_emp, pred_test)
0.8072289156626506

precision_score(y_emp, pred_test)
0.6976744186046512

recall_score(y_emp, pred_test)
0.9090909090909091
```

- **accuracy_score() =** number of correct predictions out of total predictions
- **precision_score()** = true positives / (true positives + false positives)
- **recall_score()** also known as 'Sensitivity' = true positives / (true positives + false negatives)

```
# Area Under ROC Curve
auc = roc_auc_score(y_emp, predprob_MNB[:,1])
print('AUC: %.3f' % auc)
AUC: 0.871
```

* Note : Output might be slightly different as observations are randomly assigned to train-test data.

Naive Bayes Method in Python
For Categorical Predictors

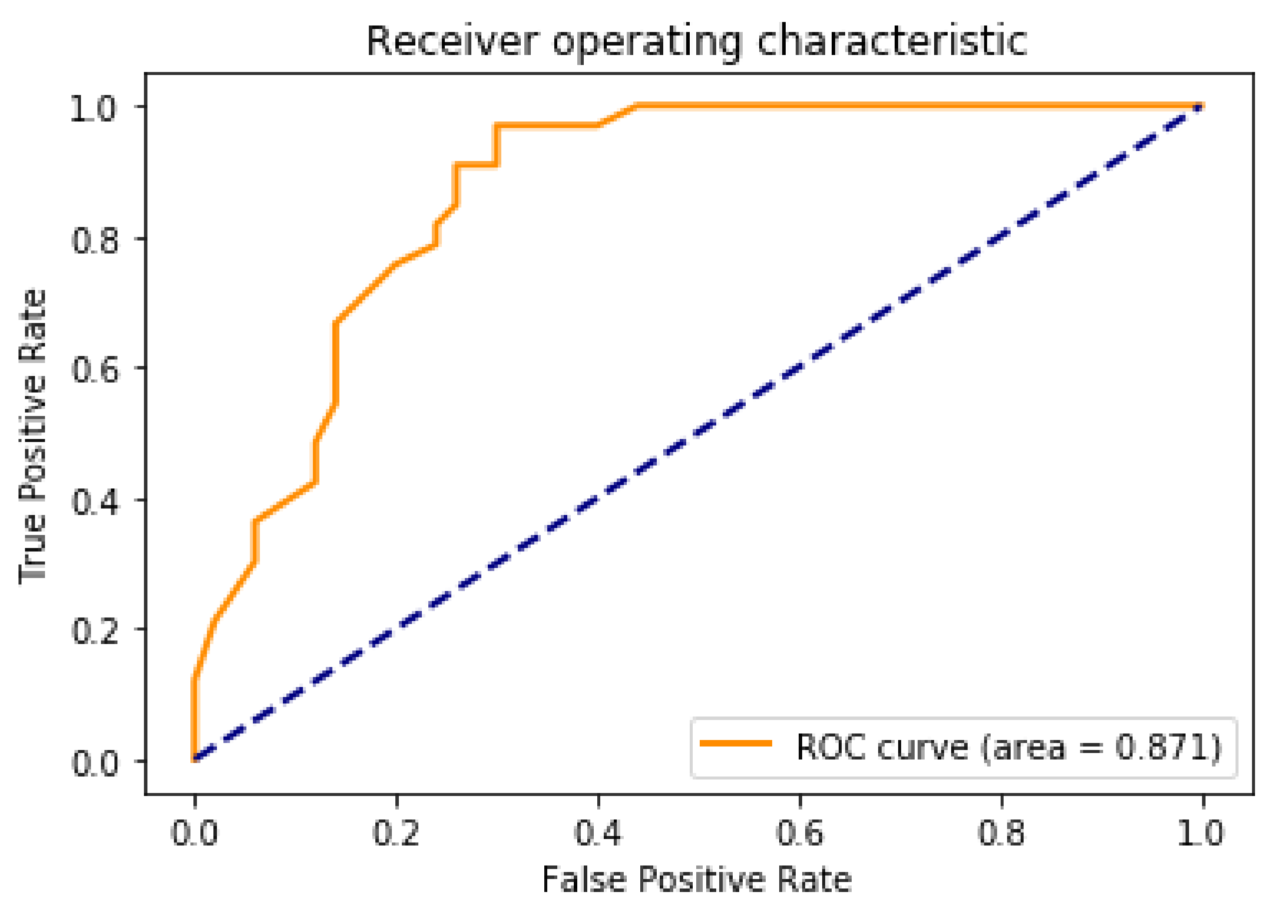
```
# ROC Curve
MNBfpr, MNBtpr, thresholds = roc_curve(y_emp, predprob_MNB[:,1])

# plot the roc curve for the model
plt.figure()
lw = 2
plt.plot(MNBfpr, MNBtpr, color='darkorange',lw=lw, label='ROC curve
(area = %0.3f)' % auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.axis('tight')
plt.xlabel('False Positive Rate');plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

DATA SCIENCE INSTITUTE

Naive Bayes Method in Python For Categorical Predictors

```
# Output
```



Predicted Probability for 1st Case

Naive Bayes Calculations (1st row)

1. $P(Y=1) = \dfrac{\text{Number of values with } Y=1}{\text{Total number of values}} = \dfrac{33}{83}$

   $P(Y=0) = \dfrac{50}{83}$

2. $P(X_1 = CS \mid Y=1) = \dfrac{18}{33}$ $\quad$ $P(X_1 = ES \mid Y=0) = \dfrac{8}{50}$

3. $P(X_2 = \text{"<3"} \mid Y=1) = \dfrac{25}{33}$ $\quad$ $P(X_2 = \text{"<3"} \mid Y=0) = \dfrac{10}{50}$

4. $P(X_3 = M \mid Y=1) = \dfrac{19}{33}$ $\quad$ $P(X_3 = M \mid Y=0) = \dfrac{27}{50}$

5. $P(X_4 = \text{external} \mid Y=1) = \dfrac{18}{33}$ $\quad$ $P(X_4 = \text{external} \mid Y=0) = \dfrac{17}{50}$

Predicted Probability for 1st Case

$$P(y=1 \mid \underline{X}) = \frac{\frac{33}{83} \times \frac{18}{33} \times \frac{25}{33} \times \frac{19}{33} \times \frac{18}{33}}{\frac{33}{83} \times \frac{18}{33} \times \frac{25}{33} \times \frac{19}{33} \times \frac{18}{33} + \frac{50}{83} \times \frac{8}{50} \times \frac{10}{50} \times \frac{27}{50} \times \frac{17}{50}}$$

$$= \boxed{0.9358}$$

Laplace Smoothing

This prob will be 0 if numerator count () is 0

Laplace smoothing will replace this probability with a value obtained by the formula:

$$\hat{\theta} i = \frac{f_i + \alpha}{N_j + \alpha d_i}$$

where

: Smoothing Parameter

: Number of observations for

: Number of classes of

DATA SCIENCE INSTITUTE

Quick Recap

Naive Bayes in Python

- `GaussianNB` for continuous variables, `MultinomialNB` for categorical variables in library `sklearn.naive_bayes`

Laplace Smoothing

- If a given class and feature value never occur together in the training data, then the frequency-based probability estimate will be zero.
- A pseudo-count is incorporated, in all probability estimates such that no probability is ever set to be exactly zero.
- This way of regularizing naive Bayes is called Laplace Smoothing

THANK YOU!

DATA SCIENCE
INSTITUTE

THANK YOU!!

DATA SCIENCE
INSTITUTE