CMSC 173 - MP 2

Instructions:

- 1. Create an overview of the problem being solved, e.g., what was the story behind the collection of the data, description of the attributes/features used,etc.
- 2. (Data Preprocessing and Exploratory Analysis) Present descriptive statistics as applicable (e.g., distribution, central tendency, variability) of the data before training the models. Clean the data if there are missing values, etc. You may perform feature engineering (i.e., creating new features out of the given features), but be sure to document your justifications.
- 3. Split your data into proportions of 70% training set and 30% testing set.
- 4. Train the following models: (a) logistic regression classifier and (b) naive Bayes classifier on the dataset.
- 5. Evaluate the performance of the trained model. You may use additional performance measures if you want, but for now I will only require the calculation of the accuracy. The accuracy measures the fraction of correct classifications. With this, you need to generate the confusion matrix. You may read this if you haven't encountered this concept before: https://www.sciencedirect.com/topics/engineering/confusion-matrix#:~:text=A%20confusion%20matrix%20represents%20the,by%20model%20as%20oth Remember to compute this matrix from the test set (not the training set).

```
In []: using Random
    using StatsBase
    using CSV
    using DataFrames
    using Plots
    using Base
    import StatisticalMeasures.ConfusionMatrices as CM

In []: dataset = CSV.read("passenger_flight.csv",DataFrame)
    Random.seed!(123)
    dataset = dataset[shuffle(axes(dataset, 1)), :]
```

Out[]: 25976×23 DataFrame

25951 rows omitted

Row	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	E C b
	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	lı
1	1	1	50	1	1	3744	5	5	
2	0	1	53	1	1	2661	4	5	
3	1	1	20	0	0	541	2	4	
4	0	1	52	0	1	944	1	2	
5	1	1	33	1	1	406	1	1	
6	0	1	51	0	0	621	2	4	
7	1	1	25	1	1	3547	2	2	
8	0	1	51	1	1	547	4	4	
9	0	1	60	0	1	438	2	4	
10	1	1	26	1	1	2085	1	1	
11	1	1	17	0	0	505	3	4	
12	0	0	22	1	0	329	3	1	
13	1	1	25	0	0	479	3	4	
:	:	:	÷	:	÷	:	:	:	
25965	0	1	27	1	1	1716	2	1	
25966	1	0	26	1	1	591	1	1	
25967	0	1	63	0	0	1024	4	4	
25968	1	1	39	1	1	2131	2	2	
25969	0	1	40	0	0	369	3	1	
25970	1	1	38	0	0	633	2	1	
25971	1	1	32	1	1	1635	2	2	
25972	0	1	43	1	1	1055	5	5	
25973	1	1	61	1	1	2273	4	4	
25974	1	1	37	1	1	695	2	4	
25975	0	1	38	0	0	1313	4	5	
25976	1	0	26	1	1	447	1	0	
4									•

Data Preprocessing

```
In []: # Replace missing values with the mean

has_missing = .!completecases(dataset)
    rows_with_missing_values = dataset[has_missing, :]
    print(rows_with_missing_values) # 83 rows have missing values in the Arrival Delay

mean_value = mean(skipmissing(dataset[:,"Arrival Delay in Minutes"]))
    transform!(dataset, All() .=> (x -> replace(x, missing => mean(skipmissing(x)))) =>
    println(dataset[1:10,:])

has_missing = .!completecases(dataset)
    rows_with_missing_values = dataset[has_missing, :]
    display(rows_with_missing_values) # no missing values
```

83×23 DataFrame

Row | Gender Customer Type Age Type of Travel Class Flight Distance Inflight wifi service Departure/Arrival time convenient Ease of Online booking Gate loca tion Food and drink Online boarding Seat comfort Inflight entertainment On-boar d service Leg room service Baggage handling Checkin service Inflight service Cl eanliness Departure Delay in Minutes Arrival Delay in Minutes satisfaction

Int64	Int64	Int64 Int6	4 Int64	Int64	Int64
Int64		Int64		Int64	Int64
Int64	Int64	Int64	Inte	54	Int64
Int64	Int64	Int64	Int64	Int64	
Union{Missing,	Int64}	Int64			

	1		1	20		0	1		2475
I	1		1	1		0	4		4
		4		2			4		2
		3		2		3	4	4	2
			sing	2	0	3		4	
1	1	IIIT2	3111g 1	55	V	0	0		265
1				1		V	3		3
		3		2			5		1
		4		1		4	J	5	_
		miss	ina	1	0	4)	
1	1	111722	1 I	51	V	1	1		3673
ı	_			5		1	5		5
		5		4			4		4
		4		5		4	-	3	-
		miss	ing	,	1	7		5	
1	1	111133	1 1	47	_	0	0		190
ı	-		_	4		0	1		3
		1		5			5		4
		5		3		4	,	5	-
		miss	ing	,	0	7		,	
1	1	111133	1 1	28	O	1	1		2556
1	_		_	3		-	3		3
		4		3			3		3
		5		3		5	_	3	
		miss	ing		1	_			
1	1		1	43	_	0	0		278
'				5			4		2
		4		3			3		4
		4		4		5		3	
		miss	ing		0				
	0		1	22		1	1		2611
'				3			3		3
		4		4			4		4
		3		5		4		4	
		miss	ing		1				
	1		1	61		1	1		1201
				2			5		2

0

missing

14

9	1	1	21 3		0 1	767 1
1 2		1 1	1 1		1 2	1
5 10 5	0	missing 1	45 4	0	1 0 4	222 4
5 5 5 0		3 5 missing	4	1	5 5	5 2
11 2	1	1	9 5		0 0 2	762 2
4 5 51		2 2 missing	1 5	0	2	4
12 1 2	1	1	62 5 2		0 0 1 2	1024 3 3
5 45		4 missing	5	0	4	2
13 4 3	0	5	65 3 2		1 0 3	705 3 3
4 16		3 missing	3	1	3	4
14 5 5	0	5	37 4 5		1 1 1 5	279 1 5
5 3 14		4 missing	1	1	2	5
15 4 4	0	4	31 4 4		1 1 4 4	3785 4 4
2 84 16	1	5 missing 1	4 45	1	5 0 0	4 2565
2	1	2	3 4		2 4	2 3
2 7 17	0	2 missing 1	4 47	0	2 1 1	4 1437
3 4	Ü	2	2		3	4
3 2 18	0	3 missing 1	4 52	0	3 1 1	1 3659
5 2 5		4 5	5 5 3		5 5 5	5 5
9 19	1	missing 1	62	1	1 1	3 2446
3 3 3		3	5 4 4		5 3 3	5 3 2
1 20	0	missing 0	33	0	1 0	587

4			0			4		4
		4	3			3		4
3 3		5	4		5		3	•
0		missing		0				
21	1	0	49		0	0		491
5			4			5		2
3 4		5	3			3	2	1
4 0		3 missing	3	1	4		3	
22	1	1 1	45	_	1	0		352
5	_	_	1		_	1		1
5		5	5			5		1
1		1	3		2		5	
26		missing		1				
23	0	1	30		1	1		1931
3 3 1		3	5 3			5 3		5 2
1		3	3		3		3	2
0		missing	,	0	,			
24	1	1	46		1	1		2475
3			3			3		3
4		4	4			3		3
5		5	4		4		4	
530 25	0	missing 0	36	1	1	1		328
3	0	0	36		1	3		2
2		3	2			2		5
5		5	5		5		2	
0		missing		0				
26	0	1	39		1	1		2475
1		-	1			3		1
5 4		5 4	4 5		4	4	5	4
0		missing	5	1	4		5	
27	0	1	21	_	0	0		632
1			5			1		4
2		1	4			2		4
2		5	4		5		2	
9	0	missing	F.0	0	1	1		0.5
28 1	0	1	50 5		1	1 5		95 5
3		3	2			2		2
3 2		1	1		2		3	
0		missing		0				
29	1	0	34		1	0		376
2		_	0			2		5
3 2		2	1 4		5	3	3	5
0		missing	4	0	5		5	
30	1	1 missing	8	J	0	0		936
		_	4		-	3		2
3 5		3	5			5		5
3		5	4		5		5	
0	•	missing	4.0	0				2455
31	0	1	42		1	1		2133
5			5			5		5

5		4	5		3		3
3		3	3		3	5	_
63		missing		1			
32 2	1	1	63 4		0 0 2		1205 4
2		2	2		2		2
2		4	4		4	2	
76		missing		0			
33	1	1	62 4		0 0		152 2
1 3		0	3		3		5
4		4	5		5	3	
8		missing		0			
34	0	1	39		1 0		125
5 5		5	1 5		1 5		1 2
2		3	5		2	5	_
131		missing		1			
35	0	1	29		1 1		3873
3		3	3		3		3 3
4		4	4		3	3	,
19		missing		0			
36	1	1	67		0 0		2446
3 3		5	4 5		3		5 4
4		4	5		5	5	4
332		missing		0	-	-	
37	0	1	55		1 1		1438
5		4	5 5		5 5		5 5
2 5		5	3		5	_	5
81						5	
20		missing		1	J	5	
38	0	missing 1	36	1	1 1		1840
	0	1	36 2	1	1 1 2		2
2	0	5	36 2 5	1	1 1 2		
	0	5 5	36 2 5 5	1	1 1 2		2
2 4 1 166 39	0	5	36 2 5 5 5		1 1 2 2 4 1 0	5	2 5 1208
2 4 1 166 39		1 5 5 missing 1	36 2 5 5 5 57 5		1 1 2 2 4 1 9 4	5	2 5 1208 4
2 4 1 166 39 5		1 5 5 missing 1	36 2 5 5 5 57 5		1 1 2 2 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	5	2 5 1208
2 4 1 166 39 5 5		5 5 missing 1 5	36 2 5 5 5 57 5		1 1 2 2 4 1 9 4	5	2 5 1208 4
2 4 1 166 39 5 5 2 27 40		1 5 5 missing 1	36 2 5 5 5 57 5 5	1	1 1 2 2 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	5	2 5 1208 4 1
2 4 1 166 39 5 5 2 27 40	1	5 missing 1 5 4 missing	36 2 5 5 57 5 5 1	1	1 1 2 2 4 1 1 0 4 1 1 0 1 1 0 3	5	2 5 1208 4 1 129 3
2 4 1 166 39 5 5 2 27 40	1	1 5 missing 1 5 4 missing 1	36 2 5 5 57 5 5 1 58 3	1	1 1 2 2 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	5	2 5 1208 4 1
2 4 1 166 39 5 5 2 27 40 5 3 5	1	5 missing 1 5 4 missing 1 5 5 5	36 2 5 5 57 5 5 1	1	1 1 2 2 4 1 1 0 4 1 1 0 1 1 0 3	5	2 5 1208 4 1 129 3
2 4 1 166 39 5 5 2 27 40 5 3 5 41	1	1 5 missing 1 5 4 missing 1	36 2 5 5 57 5 5 1 58 3 5	1	1 1 2 2 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	5	2 5 1208 4 1 129 3 5
2 4 1 166 39 5 5 2 27 40 5 3 5 3 41 4	1	5 missing 1 5 missing 1 5 missing 1 5 missing 0	36 2 5 5 57 5 5 1 58 3 5 3 40 40	1	1 1 2 2 4 1 4 1 1 1 1 1 1 1 1 1 1 1 1 1	5	2 5 1208 4 1 129 3 5
2 4 1 166 39 5 5 2 27 40 5 3 5 3 41 4	1	5 missing 1 5 missing 1 5 missing 0	36 2 5 5 5 57 5 5 1 58 3 5 3 40 4	1	1 1 2 2 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	5	2 5 1208 4 1 129 3 5
2 4 1 166 39 5 5 2 27 40 5 3 5 3 41 4 4 5	1	5 missing 1 5 missing 1 5 missing 0	36 2 5 5 57 5 5 1 58 3 5 3 40 4	1 1	1 1 2 2 4 1 4 1 1 1 1 1 1 1 1 1 1 1 1 1	5	2 5 1208 4 1 129 3 5
2 4 1 166 39 5 5 2 27 40 5 3 5 3 41 4	1	5 missing 1 5 missing 1 5 missing 0	36 2 5 5 57 5 5 1 58 3 5 3 40 4	1	1 1 2 2 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	5 5	2 5 1208 4 1 129 3 5
2 4 1 166 39 5 5 2 27 40 5 3 5 3 41 4 4 5 193	1 0	5 missing 1 5 missing 1 5 missing 0 4 missing 0	36 25 5 57 57 5 1 58 3 5 3 40 40 4 3	1 1	1 1 2 2 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	5 5	2 5 1208 4 1 129 3 5

5	5	2		5	2
254	missing		1		_
	1 1	29		1 1	3772
5 5	4	5 5		5 5	5 3
5 3	5	3		4	5
0	missing		1		
'	0 0	15		1 0	544
4 5	4	0 5		4 5	3 3
4	2	1		4	5
58	missing		0		
	0 1	59		1 1	164
0 4	5	5 3		0 5	5 5
5	5	1		5	1
81	missing		1		
'	0 1	14		1 1	2515
5 4	4	5 4		5 4	5 4
2	4	5		4	4
8	missing		1		
	0 1	34		1 1	237
5 3	5	5 5		5 4	5 4
4	4	5		4	3
71	missing		1		
	0 1	53		0 0	480
3 2	3	4 4		3 3	3 3
3	5	2		3	1
0	missing		0		
'	0 0	21		1 0	675
4	4	2		4 2	3 2
2 3 0	3	4		3	2
	missing		0		
	0 1	44		1 0	598
4 2	4	5 1		5 5	5 5
4	5	3		5	5
17	missing		1		
51 4	0 0	21 4		1 0 4	1246 3
3	4	2		3	4
3 5	5	5		5	3
106	missing		1		
52 1	0 1	41 1		1 1 1	3758
2	5	5		5	1 5
5	5	4		5	5
23	missing		1		
	1 1	42 2		1 0 4	383 4
3 3 2	3	3		3	1
2	3	1		3	3

54 0 0 24 1 1 627 1 4 4 4 2 1 4 3 4 4 0 missing 0 0 0 55 0 1 58 1 1 927 5 5 5 5 5 5 5 5 5 1 1 5 2 5 4 3 3 3 3 3	72				0				
1	73 54	а	missing	2/1	0	1	1		627
4 1 4 4 2 0 missing 0 0 927 55 0 1 58 1 1 927 5 5 5 5 5 5 5 1 1 5 5 5 5 5 5 13 missing 1 1 3 399 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3		0	O			_			
1 4 3 4 4 0 missing 0 927 5 5 5 5 5 1 1 5 5 5 1 1 5 5 5 13 missing 1 0 0 399 4 4 4 4 4 4 4 4 3			1						
55 0 1 58 1 1 927 5 4 1 1 3 399 4 3				3		4		4	
5 1 1 5 5 5 5 5 5 5 5 5 5 4 1 3 3 3 3 3 3 3 3 4	0		missing		0				
1 5 5 2 5 4 13 missing 1 399 4 5 4 4 4 56 1 1 7 0 0 399 4 4 4 4 4 4 6 0 1 4 4 4 8 1 26 1 1 2417 3 3 3 3 1 26 1 1 2417 3 4 1		0	1			1			
5 5 2 5 4 13 missing 1 399 4 3 4 1 1 4 4 4									
13									5
56 1 1 7 0 0 399 4 4 4 4 4 3 1 1 4 4 6 1 1 2417 3 1 26 1 1 2417 3 3 3 3 3 3 4 4 4 4 3 3 4 5 1 5 4 4 8 1 0 28 1 0 636 3 3 2 2 1 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 1 4 4 2 1 4 4 2 1 4 4 2 1 4 4 2 1 4 4 4 4 4				2		5	i	4	
4 4 4 4 4 4 4 4 4 4 3 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		4		7	1	0	0		200
4 4 4 4 4 4 4 6 6 1 1 4 4 4 4 6 9 1 26 1 1 2417 3 4 2 2 2 2 <td></td> <td>T</td> <td>1</td> <td></td> <td></td> <td>0</td> <td></td> <td></td> <td></td>		T	1			0			
3 1 1 4 4 6 missing 0 1 26 1 1 2417 3 1 26 1 1 2417 3 2 1 1 4 4 2 2 1 1 4 2 2 1 1 4 2 2 1 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2			1						
0 missing 0 1 26 1 1 2417 3 1 26 1 1 3 3 4 4 4 4 3 3 4 5 1 5 4 636 3 1 0 28 1 0 636 3 2 2 1 0 636 3 1 3 2 1 0 636 3 1 3 2 1 1 3 3 3 3 3 3 3 3 3 3 3 3 4 2 2 1 4 4 2 2 1 4 4 2 2 1 4						Δ		4	4
57 0 1 26 1 1 2417 3 4 4 4 3 3 4 5 1 5 4 3 6 missing 1 0 636 3 3 3 3 58 1 0 28 1 0 636 3 2 0 1 4 2 2 1 4 4 2 2 1 4 4 2 2 1 4 4 2 2 1 4	0			-	0	_		7	
3 4 4 4 3 3 4 4 4 4 3 6 missing 1 5 4 58 1 0 28 1 0 636 3 3 2 2 2 1 4 3 1 3 2 2 69 1 1 31 0 0 1325 4 3 3 3 4 1 0 59 1 1 31 0 0 1325 4 2 2 4 4 5 3 3 4 1 0 0 1325 4 2 2 1 4 2 2 1 4 4 2 2 1 4 4 2 2 1 4<		0		26		1	1		2417
4 4 4 4 4 3 4 5 1 5 4 6 58 1 0 28 1 0 636 3 3 3 3 3 3 2 3 2 2 1 4 4 3 1 3 2 1 69 1 1 31 0 0 1325 4 5 1 4 2 1 4 5 1 4 1 60 0 1 46 1 1 1521 2 2 2 2 2 2 2 2 4 2 2 2 4 4 4 4 4 4 2 2 4 2 2 2 4 4 4 4 4 4 2 2 4 2 2 2 4 4 3 5 5<									
0 missing 1 0 28 1 0 636 3 3 2 2 1 4 3 1 3 2 59 1 1 31 0 0 1325 4 2 5 4 2 2 1 4 5 4 1 4 5 1 4 5 1 4 1 4 1 4 4 1 4			4				4		
58 1 0 28 1 0 636 3 3 2 2 1 4 3 1 3 2 1 60 missing 0 0 1325 4 1 31 0 0 1325 4 2 1 4 2 1 4 5 1 4 1 6 0 1 46 1 1 1 1521 2 2 2 2 2 2 2 4 4 2 2 2 2 2 4 4 2 3 3 <td< td=""><td>4</td><td></td><td>5</td><td>1</td><td></td><td>5</td><td></td><td>4</td><td></td></td<>	4		5	1		5		4	
3					1				
2		1	0			1			
4 3 1 3 2 60 missing 0 0 1325 4 5 4 2 1 4 5 1 4 5 3 3 4 1 60 0 1 46 1 1 1521 2 3 4 2 2 2 2 3 4 2 2 2 4 1 39 1 1 1576 4 4 5 5 5 5 5 5 3 5 5 5 61 1 1 39 1 1 1576 4 4 5 5 5 5 5 5 5 3 5 5 5 5 62 1 0 38 1 1 759 3 5 5 4 4 4 4 0 missing 1	3		_						
0 missing 0 0 0 1325 4 5 4 2 1 4 5 1 4 2 5 3 3 4 1 4 6 1 4 1 1 1521 1 1 1521 2 2 4 2 3									1
59 1 1 31 0 0 1325 4 2 1 4 2 1 4 5 1 4 2 5 3 3 4 1 4 60 0 1 46 1 1 1521 2 3 4 2 2 2 2 3 4 2 2 2 4 4 2 2 2 2 4 3 4 2 2 2 2 4 4 2 3 3 3				1	0	3	}	2	
4		1		21	0	0	0		1225
1 4 5 1 4 1 5 3 3 4 1 1 60 0 1 46 1 1 1521 2 4 4 4 4 4 2 3 4 2 2 2 2 2 4 2 2 2 4 missing 0 1 1 1576 4 4 39 1 1 1576 4 4 4 4 4 4 2 4 5 5 5 5 5 5 5 3 5 5 5 5 6 1 0 38 1 1 759 3 3 4 4 4 3 6 1 0 38 1 1 7 79 3 5 5 4 4 4 3 1 1 2 4 4		1	1			V			
5 3 3 4 1 0 missing 0 0 1 1521 60 0 1 46 1 1 1521 2 4 4 4 4 4 2 2 2 2 2 2 2 2 2 4 2 2 2 4 missing 0 1 1 1576 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 3 5 5 5 5 6 1 1 0 38 1 1 759 3 3 4 4 4 3 1 4 4 4 3 6 1 1 2 4 4 4 4 4 4 6 1 1 2 3 1 1 2 3 3 3 3			4						
0 missing 0 60 0 1 46 1 1 1521 2 3 4 2 2 2 2 2 4 2 2 2 4 missing 0 1 1 1576 4 4 39 1 1 1576 4 4 5 5 5 5 5 5 3 5 5 5 6 1 1 0 38 1 1 759 3 3 3 3 1 759 3 5 5 4 4 3 4 3 4 4 4 4 9 1 2 4 4 4 1 1 20 0 0 1050 1 1 3 3 3 3 3 1 3 3 3 3 63 0 1	5					4		1	
60 0 1 46 1 1 1521 2 3 4 2 2 2 2 2 4 2 2 2 4 missing 0 0 1 1 1576 4 3 1 1 759 3 1 1 759 3 1 1 759 3 1 1 759 3 1 1 759 3 1 1 759 3 1 1 759 3 1 1 759 3 1 1 1 1 1 1 1 1 1 2 3 1 1 1 </td <td>0</td> <td></td> <td></td> <td></td> <td>0</td> <td></td> <td></td> <td></td> <td></td>	0				0				
2 3 4 2 2 2 4 2 2 4 1 1 1 1576 4 1 1 1 39 1 1 1 1576 4 2 4 5 5 5 5 6 5 5 5 5 5 5 5 5 5 5 5 5 5 5	60	0	1	46		1	1		1521
2	2								
4 missing 0 61 1 1 39 1 1 1576 4 4 4 4 4 4 2 4 5 3 5 5 5 5 5 3 5 5 5 5 60 1 0 38 1 1 759 3 3 4 3 4 3 1 4 3 4 4 3 4 3 4 4 3 63 0 1 20 0 0 1050 1050 1 2 4 4 3 <t< td=""><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>2</td></t<>									2
61 1 1 39 1 1 1 1576 4	2			4	_	2	<u>}</u>	2	
4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 7				20	0	4	4		4576
2 4 5 3 5 5 5 60 missing 1 1 759 62 1 1 0 38 1 1 759 3 3 4 4 4 3 1 4 3 4 4 4 4 4 9 missing 1 0 0 1050 1050 1 2 3 1 20 0 0 1050 1 2 3 1 3 3 3 3 3 3 9 1 3 4 4 4 3 3		T	1			1			
5 5 3 5 5 0 missing 1 1 759 3 38 1 1 759 3 3 4 3 3 1 4 3 4 4 4 3 3 5 5 4 4 4 0 missing 1 1 2 1 1 2 3 1 20 0 0 0 1050 1050 1 2 4 3 4 1 1 </td <td></td> <td></td> <td>4</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>			4						
0 missing 1 62 1 0 38 1 1 759 3 3 3 1 3 1 4 3 4 4 4 3 3 5 5 4 4 4 63 0 1 20 0 0 1050 1 2 4 3 3 3 3 3 1 3 3 3 3 3 3 5 4 4 4 3 3 64 0 1 52 1 1 874 1 1 5 5 5 5 5 5 5 5 5 5 5 5	5					5		5	,
62 1 0 38 1 1 759 3 3 4 4 3 1 4 3 4 4 4 4 60 missing 1 0 0 0 1050 1 4 4 3 3 3 3 3 1 3 3 3 3 3 3 5 4 4 4 3 3 64 0 1 52 1 1 874 1 1 5 5 5 5 5 5 5 4 5 5 5 5 5 5 4 5 5 5 5	0				1				
3 3 4 4 3 3 3 3 3 3 3 3 3 4 4 3 3 4 4 4 4 4 4 4 4 4 4 4 60 1050 1050 1 1 2 2 1 1 2 2 3 4 4 4 3 3 3 4 4 4 3 3 4 4 4 3 4 4 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1<		1		38		1	1		759
3 5 4 4 0 missing 1 63 0 1 20 0 0 1050 1 4 1 2 3 1 3 3 3 3 3 5 4 4 3 3 0 missing 0 0 1 874 1 1 5 1 1 874 1 5 5 5 5 5 5 5 5 5 5 5	3						3		
0 missing 1 63 0 1 20 0 0 1050 1 4 4 1 2 3 1 3 3 3 3 64 0 1 52 1 1 874 1 <				4			4		3
63 0 1 20 0 0 1050 1 4 1 2 3 3 3 3 1 3 4 3 4 3 0 missing 0 64 0 1 52 1 1 874 1 2 5 5 5 5 5 5 5 5 5 4 5 5 5	3			5		4		4	
1 4 1 2 3 1 3 3 3 3 5 4 4 3 64 0 1 52 1 1 874 1					1				
3 5 4 4 3 0 missing 0 0 64 0 1 52 1 1 874 1 1 1 1 1 1 1 2 5 5 5 5 5 5 5 5 4 5 5 5	63	0	1			0			
3 5 4 4 3 0 missing 0 0 64 0 1 52 1 1 874 1 1 1 1 1 1 1 2 5 5 5 5 5 5 5 5 4 5 5 5	1		4						
0 missing 0 64 0 1 52 1 1 1 874 1 1 1 1 1 2 5 5 5 5 5 5 5	<u>ა</u>					л		2	3
64 0 1 52 1 1 874 1 1 1 1 1 2 5 5 5 5 5 5 4 5 5	э 0			4	а	4	•	Э	
1 1 1 1 2 5 5 5 5 5 5 4 5 5		a		52	J	1	1		874
2 5 5 5 5 5 5 4 5 5		J	-			-			
5 5 4 5 5	2		5						
10 missing 1	5		5			5		5	
			missing		1				

65	0	1	64		0	0		1226
4	Ü	_	4		Ü	3		4
2		3	4			2		2
3 0		2 missing	4	0	2		4	
66	0	1	59	O	1	0		547
4			1			1		1
5 4		2	3			4	4	4
3		4 missing	4	1	4		4	
67	1	1	58	_	0	0		618
1 2			4			1		2
2 4		1 5	2 3		5	2	2	4
4 48		o missing	3	0	5		2	
68	0	0	37		1	1		1428
2		_	2			2		3
1 4		2 4	1 5		4	1	1	4
85		missing	,	0	4		1	
69	0	0	22		1	1		1035
5 5 2		_	4			5		_1
5 2		5 5	2 4		4	5	5	5
0		missing	4	1	-		,	
70	0	1	53		1	1		813
5 2			5			1		5
2		4 2	5 5		2	2	3	2
41		missing		1	_		,	
71	1	1	9		0	0		427
1 4		1	0 4			1		1 4
5		1	5		1	4	4	4
0		missing		0				
72	0	1	64		0	0		622
3 3 3		3	3 4			3		5 1
3		1	2		1	_	5	_
109		missing		0				
73	1	1	26		1	0		645
5 5		5	2 5			2 5		2 1
3		4	5		2		5	_
11		missing		1				
74 1	0	1	27 1		1	1 5		2192 1
2		2	2			2		3
3		4	5		5		2	
42 75	4	missing	22	1	4	1		2.42
75 1	1	1	32 1		1	1 1		240 1
5		5	5			5		3
		5	4		4		5	
70 76	α	missing	4.4	1	e	0		EEO
76	0	1	44		0	0		550

4				2				1	4
1 4		3		2 3			4	1	4 4
1		4		3			4		2
51			sing		0				
77	0		1	25		0		1	557
3				4				4	3
4		4		4			4		3
1		3	_	2			5		4
32	4	mis	sing	20	0	4		4	072
78	1		0	29		1		1 2	972 2
2		2		2 3			3	2	5
4		5		<i>5</i>			5		3
2		miss	ing	5	0		J		5
79	0	111111	0	26	Ü	1		0	110
1	· ·			0		_		1	5
5		1		5			5		1
5		3		4			2		5
0		miss	ing		0				
80	1		1	41		1		1	1040
5				5				5	5
3		2		2			5		5
5		5 .		4			5		3
0	4	miss	_	25	1	4		0	1017
81 3	1		0	25 4		1		0 4	1017 5
5		4		5			5	4	2
5		2		3			2		5
126			ssing	,	0		_		,
82	1		1	62		0		0	432
2				3				2	4
4		2		4			4		3
4		5		3			4		4
0		miss			0				
83	1		0	26		1		1	447
1				0				1	4
4		1		4			- 4		5
4		4	ina	4	01022	DataF	5		4
	ender	miss Customer	_	Λαο	010×23		-	Class	Elight Distance Tu
				_					Flight Distance In Online booking Gate
									ht entertainment On-
									ice Inflight service
									es satisfaction
		Float64	-		Float6		-		Float64 Fi
oat64			at64					Float64	
t64	Floa	t64	Floa	t64	F1	oat64		Float	64 F.
oat64		Float64		Floa			Float	t64	Float64
Float64	Fl	oat64			Float	64			Float64
1	1.0		1.0	50.0		1	.0	1.0	3744.0
5.0				5.0				5.0	5.0

2.0							•		
3.0		4.0	4			4.0		2.0	4.0
4.0		4.0		5.0		4.0		3.0	
0.0									
'	0.0		1.0	53.0			1.0		2661.0
4.0				5.0			5.0		5.0
3.0		1.0	2	.0		4.0			4.0
3.0		4.0		4.0		4.0		2.0	
6.0			8.0		0.0				
3	1.0		1.0	20.0			0.0		541.0
2.0				4.0			2.0		3.0
4.0		2.0	4	.0		4.0	9		2.0
2.0		4.0		2.0		3.0		4.0	
38.0			38.0		0.0				
4	0.0		1.0	52.0		0.0	1.0		944.0
1.0				2.0			1.0		2.0
2.0		3.0	2	.0		2.0	9		2.0
1.0		2.0		1.0		2.0		2.0	
34.0					0.0				
5	1.0			33.0		1.0	1.0		406.0
1.0				1.0			1.0		1.0
4.0		4.0	4			4.0			3.0
5.0		4.0		3.0			-	4.0	
0.0			0.0	3.0		3.0			
6	0.0			51.0	1.0	a a	0.0		621.0
2.0	0.0		1.0	4.0			2.0		1.0
2.0		4.0	1	0			2.0		3.0
2.0		3.0	7	5.0			9	3 0	3.0
0.0		5.0	a a	5.0		3.0		5.0	
	1.0			25.0	0.0	1 0	1.0		3547.0
2.0	1.0		1.0	2.0		1.0	2.0		2.0
5.0		5.0	5			5.0			
5.0		1.0)	4.0		4.0		5.0	5.0
		1.0	0.0			4.0		5.0	
0.0	0 0		0.0		1.0	1.0	1.0		F 47 0
	0.0		1.0			1.0			547.0
4.0		4.0	_	4.0		4	4.0		4.0
2.0		4.0	5	.0		4.0	0		4.0
4.0		4.0		3.0		4.0		5.0	
0.0			0.0		1.0				
9	0.0		1.0	60.0		0.0			438.0
2.0				4.0			2.0		3.0
2.0		4.0	4	.0		5.0	9		5.0
2.0		5.0		5.0		5.0		3.0	
0.0			0.0		0.0				
10	1.0		1.0	26.0		1.0	1.0		2085.0
1.0				1.0			1.0		1.0
5.0		5.0	5	.0		5.0	9		4.0
5.0		5.0		3.0		4.0		5.0	
37.0			23.0		1.0				

0×23 DataFrame

```
Type
                                                                 Inflight
                                                                         Departure/Arrival
                    Customer
                                                        Flight
       Row Gender
                                       of
                                               Class
                               Age
                                                                 wifi
                                                        Distance
                                                                         time convenient
                     Type
                                       Travel
                                                                 service
            Float64 Float64
                               Float64 Float64 Float64
                                                                 Float64 Float64
      4
In [ ]: # rename column names
        col_names = names(dataset)
        new_col_names = map(lowercase, String.(col_names)) # convert to lower case
        new_col_names .= replace.(new_col_names, " "=>"_", "-"=>"", "/"=>"_") # replace spa
        rename!(dataset, new_col_names)
        display(dataset)
```

25976×23 DataFrame 25951 rows omitted

Row	gender	customer_type	age	type_of_travel	class	flight_distance	inflight_v
	Float64	Float64	Float64	Float64	Float64	Float64	Float64
1	1.0	1.0	50.0	1.0	1.0	3744.0	
2	0.0	1.0	53.0	1.0	1.0	2661.0	
3	1.0	1.0	20.0	0.0	0.0	541.0	
4	0.0	1.0	52.0	0.0	1.0	944.0	
5	1.0	1.0	33.0	1.0	1.0	406.0	
6	0.0	1.0	51.0	0.0	0.0	621.0	
7	1.0	1.0	25.0	1.0	1.0	3547.0	
8	0.0	1.0	51.0	1.0	1.0	547.0	
9	0.0	1.0	60.0	0.0	1.0	438.0	
10	1.0	1.0	26.0	1.0	1.0	2085.0	
11	1.0	1.0	17.0	0.0	0.0	505.0	
12	0.0	0.0	22.0	1.0	0.0	329.0	
13	1.0	1.0	25.0	0.0	0.0	479.0	
:	:	:	:	:	:	:	
25965	0.0	1.0	27.0	1.0	1.0	1716.0	
25966	1.0	0.0	26.0	1.0	1.0	591.0	
25967	0.0	1.0	63.0	0.0	0.0	1024.0	
25968	1.0	1.0	39.0	1.0	1.0	2131.0	
25969	0.0	1.0	40.0	0.0	0.0	369.0	
25970	1.0	1.0	38.0	0.0	0.0	633.0	
25971	1.0	1.0	32.0	1.0	1.0	1635.0	
25972	0.0	1.0	43.0	1.0	1.0	1055.0	
25973	1.0	1.0	61.0	1.0	1.0	2273.0	
25974	1.0	1.0	37.0	1.0	1.0	695.0	
25975	0.0	1.0	38.0	0.0	0.0	1313.0	
25976	1.0	0.0	26.0	1.0	1.0	447.0	
4							+

Detection of Outliers and Removal

```
In [ ]: q1 = []
        q3 = []
        col names = names(dataset)
        for i in col_names
            push!(q1, quantile(dataset[:,i], 0.25))
            push!(q3, quantile(dataset[:,i], 0.75))
        end
        iqr_val = q3-q1
        lower_bound = q1 - 1.5 * iqr_val
        upper_bound = q3 + 1.5 * iqr_val
        outlier = BitVector()
        for row in eachrow(dataset)
            is_outlier = 0
            for col_idx in 1:length(col_names)
                if row[col_idx] < lower_bound[col_idx] || row[col_idx] > upper_bound[col_id
                    is_outlier = 1
                end
            end
            push!(outlier, is_outlier)
        end
        cleaned_dataset = dataset[.!outlier,:]
        println("Number of rows before removal: ", size(dataset)[1])
        println("Number of rows after removal: ", size(cleaned_dataset)[1])
       Number of rows before removal: 25976
       Number of rows after removal: 15234
In [ ]: # split dataframe into 2 df depending on pct
        function splitdf(df, pct)
            @assert 0 <= pct <= 1
            ids = collect(axes(df, 1))
            shuffle!(ids)
            sel = ids .<= nrow(df) .* pct
            train = view(df, sel, :)
            test = view(df, .!sel, :)
            # println(hcat(train[:,1:end-1], DataFrame("satisfaction"=>train[:,end])) == tr
            return train[:,1:end-1], DataFrame("satisfaction"=>train[:,end]), test[:,1:end-
        end
        (x_train, y_train, x_test, y_test) = splitdf(cleaned_dataset, 0.7)
```

Out[]: (10663×22 DataFrame

Row	gender Float64	<pre>customer_type Float64</pre>	age Float64	<pre>type_of_travel Float64</pre>	class Float64	flight_dist Float64
1	0.0	1.0	53.0	1.0	1.0	26
2	1.0	1.0	33.0	1.0	1.0	4
3	0.0	1.0	51.0	0.0	0.0	6
4	1.0	1.0	25.0	1.0	1.0	35
5	0.0	1.0	51.0	1.0	1.0	5
6	0.0	1.0	60.0	0.0	1.0	4
7	1.0	1.0	17.0	0.0	0.0	5
8	0.0	1.0	24.0	0.0	0.0	6
9	0.0	1.0	31.0	1.0	1.0	35
10	1.0	1.0	53.0	0.0	0.0	4
11	1.0	1.0	57.0	0.0	0.0	8
:	:	:	:	:	:	: ×
10654	1.0	1.0	40.0	0.0	0.0	8
10655	0.0	1.0	31.0	1.0	1.0	11
10656	0.0	1.0	53.0	0.0	0.0	2
10657	0.0	1.0	32.0	1.0	1.0	12
10658	0.0	1.0	57.0	1.0	1.0	11
10659	0.0	1.0	27.0	1.0	1.0	17
10660	1.0	1.0	38.0	0.0	0.0	6
10661	1.0	1.0	32.0	1.0	1.0	16
10662	1.0	1.0	37.0	1.0	1.0	6
10663	0.0	1.0	38.0	0.0	0.0	13

17 columns and 10642 rows omitted,

10663×1	DataFrame
---------	-----------

Row	satisfaction Float64		
1	0.0		
2	1.0		
3	0.0		
4	1.0		
5	1.0		
6	0.0		
7	0.0		
8	0.0		
9	1.0		
10	0.0		
11	0.0		
:	:		
10654	0.0		
10655	1.0		
10656	0.0		
10657	0.0		
10658	1.0		
10659	0.0		
10660	0.0		
10661	1.0		
10662	0.0		
10663	1.0		

10642 rows omitted, **4571×22 DataFrame**

Row	gender	customer_type	age	type_of_travel	class	flight_dista	•••
	Float64	Float64	Float64	Float64	Float64	Float64	•••

1	1.0	1.0	64.0	0.0	0.0	29
2	0.0	1.0	70.0	1.0	1.0	11
3	0.0	1.0	48.0	0.0	0.0	73
4	0.0	1.0	47.0	0.0	0.0	48
5	0.0	1.0	39.0	1.0	1.0	8
6	1.0	1.0	39.0	1.0	1.0	273
7	0.0	1.0	47.0	0.0	0.0	30
8	0.0	1.0	44.0	1.0	1.0	89
9	1.0	1.0	30.0	1.0	0.0	94 …
10	1.0	1.0	10.0	0.0	1.0	118
11	0.0	1.0	28.0	1.0	1.0	104
:	:	:	:	:	:	i N
4562	0.0	1.0	58.0	1.0	1.0	211
4563	1.0	1.0	38.0	1.0	0.0	22
4564	0.0	1.0	53.0	1.0	1.0	250
4565	0.0	1.0	54.0	1.0	1.0	32
4566	0.0	1.0	39.0	1.0	1.0	124
4567	0.0	1.0	23.0	0.0	1.0	172
4568	0.0	1.0	38.0	1.0	0.0	21
4569	1.0	1.0	36.0	1.0	0.0	19
4570	1.0	1.0	45.0	1.0	1.0	44
4571	1.0	1.0	44.0	1.0	1.0	130

17 columns and 4550 rows omitted,

4571×1 Row	DataFrame satisfaction Float64
1	0.0
2	0.0
3	0.0
4	0.0
5	1.0
6	1.0
7	0.0
8	1.0
9	0.0
10	0.0
11	1.0
:	:
4562	1.0
4563	1.0
4564	1.0
4565	1.0
4566	1.0
4567	0.0
4568	0.0
4569	1.0
4570	1.0
4571	0.0

4550 rows omitted)

Naive Bayes

```
In [ ]: # build conditional probability table
        cont_col_names = ["age", "flight_distance", "departure_delay_in_minutes", "arrival_
        disc_col_names = [name for name in names(dataset) if name ∉ cont_col_names && name≠
        train = hcat(x train, y train)
        # calculate discrete probabilities
        function count_disc_prob(df, col_name)
            return combine(groupby(df, [col_name, "satisfaction"]), nrow)
        end
        cond_prob_table = Dict()
        for name in disc_col_names
            cond_prob_table[name] = count_disc_prob(train, name)
        end
        # calculate continuous probabilities
        function count_cont_prob(df, col_name)
            a = combine(groupby(df, "satisfaction"), [col_name] => mean, [col_name] => std)
        end
        for name in cont col names
            cond_prob_table[name] = count_cont_prob(train, name)
        end
In [ ]: # calculate likelihood for continuous data
        function likelihood(cond_prob_table, feature, satisfaction, x)
            feature_table = cond_prob_table[feature]
            prob_values = filter(row -> row.satisfaction == satisfaction, feature_table)
            # get mean and variance
            \mu = prob_values[1,2]
            \sigma = prob_values[1,3]
            return (1/(\sigma * sqrt(2\pi))) * exp((-1/2) * ((x-\mu)/\sigma)^2)
        end
        # calculate probabilities for discrete (categorical) data
        function disc_cond_prob(cond_prob_table, feature, satisfaction, x)
            feature_table = cond_prob_table[feature]
            feature_table = filter(row -> row.satisfaction==satisfaction, feature table)
            total = sum(feature_table[:,:nrow])
            val = 0
            try
                val = filter(row -> row[feature] == x, feature_table)[1,end]
            catch
                val = 0
            end
            # apply laplace smoothing
            return (val+1)/(total+1)
        end
        # run test
        function test(x_test)
```

```
predictions = []
            # iterate all training data
            for i in 1:size(x_test)[1]
                test_case = x_test[i,:]
                p_satisfied_proportional = 1
                p_not_satisfied_proportional = 1
                # get probabilities of all features
                for col_name in names(test_case)
                    # treat discrete and continuous features separately
                    if col_name ∈ disc_col_names
                         p satisfied proportional *= disc cond prob(cond prob table, col nam
                         p_not_satisfied_proportional *= disc_cond_prob(cond_prob_table, col
                    else
                         p_satisfied_proportional *= likelihood(cond_prob_table, col_name, 1
                         p_not_satisfied_proportional *= likelihood(cond_prob_table, col_nam
                    end
                end
                # calculate probabilities
                p_satisfied = (p_satisfied_proportional / (p_satisfied_proportional+p_not_s
                p_not_satisfied = (p_not_satisfied_proportional / (p_satisfied_proportional
                # count correct and incorrect predictions
                if p satisfied >= 0.5
                    push!(predictions, 1)
                else
                    push!(predictions, 0)
                end
            end
            return predictions
        end
Out[ ]: test (generic function with 2 methods)
```

```
In [ ]: y_pred = test(x_test)
        cm = CM.confmat(y_pred, y_test[:,"satisfaction"])
        display(cm)
        println("Total correct predictions: ", (cm(0,0) + cm(1,1)))
        println("Total incorrect predictions: ", (cm(1,0) + cm(0,1)))
        println("Total test rows: ", size(y_test)[1])
        println("Model accuracy: ",(cm(0,0) + cm(1,1)) / size(y_test)[1])
```

	Ground	Truth
 Predicted	0.0	1.0
0.0	2005	279
1.0	191	2096

Total correct predictions: 4101 Total incorrect predictions: 470

Total test rows: 4571

Model accuracy: 0.8971778604244148