OneR

October 18, 2021

The Restaurant Decision

Decide whether to wait for a table at a restaurant, based on the following atributes:

- Choice: is there an alternate restaurant nearby?
- Bar: is there a comfortable bar area to wait in?
- Day: is today Friday or Saturday?
- Hungry: are we hungry?
- Patron: how many people are in the restaurant?
- Price: what's the price range?
- Rain: is it raining outside?
- Booking: have we made a reservation?
- Type: what kind of restaurant is it?
- Time: what's the estimated waiting time?

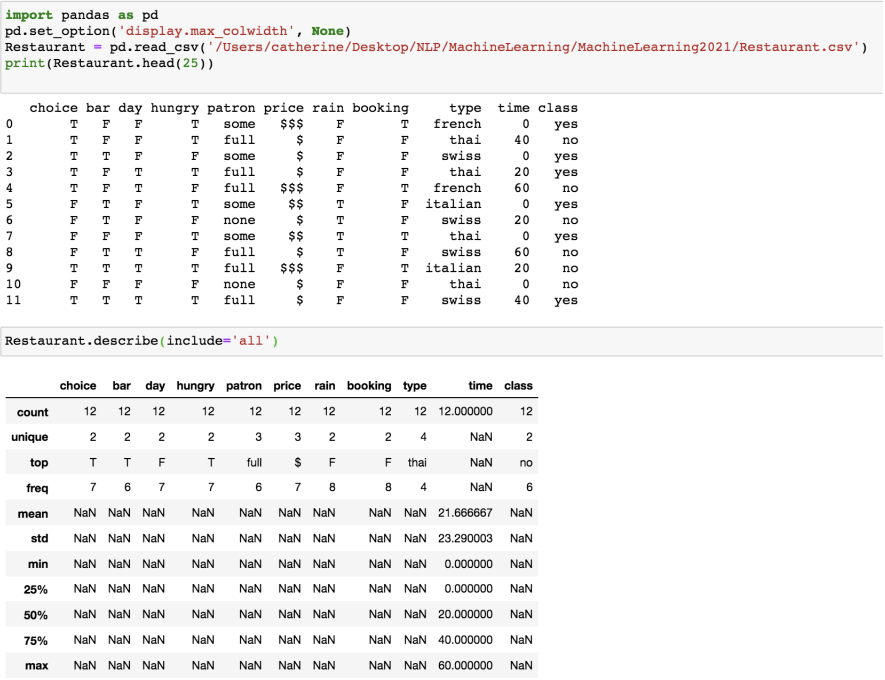
Restaurant data

choice	bar	day	hungry	patron	price	rain	booking	type	time	**wait**
T	F	F	T	some	$$$	F	T	french	0	yes
T	F	F	T	full	$	F	F	thai	40	no
T	T	F	F	some	$	F	F	swiss	0	yes
T	F	T	T	full	$	F	F	thai	20	yes
T	F	T	F	full	$$$	F	T	french	60	no
F	T	F	T	some	$$	T	F	italian	0	yes
F	T	F	F	none	$	T	F	swiss	20	no
F	F	F	T	some	$$	T	T	thai	0	yes
F	T	T	F	full	$	T	F	swiss	60	no
T	T	T	T	full	$$$	F	T	italian	20	no
F	F	F	F	none	$	F	F	thai	0	no
T	T	T	T	full	$	F	F	swiss	40	yes

In this sample, we have 12 instances. Classification: "wait": with 6 "no" and 6 "yes". Choose the most frequent to be the default rule.

OneR

Attributes	possible values	Rules	Erros	Total Error
Choice	T=7, wait(T=4, F=3)	T → T	3/7	5/12
	F=5, wait(T=2, F=3)	F → F	2/5	
Bar	T=6, wait(T=3, F=3)	T → T	3/6	6/12
	F=6, wait(T=3, F=3)	F → F	3/6	
Fri/Sat	T=5, wait(T=2, F=3)	T → F	2/5	5/12
	F=7, wait(T=4, F=3)	F → T	3/7	
Hungry	T=7, wait(T=5, F=2)	T → T	2/7	3/12
	F=5, wait(T=1, F=4)	F → F	1/5	
Patrons	**Some**=4, wait(T=4, F=0)	**Some → T**	**0/4**	**2/12**
	Full=6, wait(T=2, F=4)	**Full → F**	**2/6**	
	None=2, wait(T=0, F=2)	**None → F**	**0/2**	
Price	$$$ =3, wait(T=1, F=2)	$$$ → F	1/3	4/12
	$$=2, wait(T=2, F=0)	$$ → T	0/2	
	$=7, wait(T=3, F=4)	$ → F	3/7	
Rain	T=4, wait(T=2, F=2)	T → T	2/4	6/12
	F=8, wait(T=4, F=4)	F → F	4/8	
Booking	T=4, wait(T=2, F=2)	T → T	2/4	6/12
	F=8, wait(T=4, F=4)	F → F	4/8	
Type	French=2, wait(T=1, F=1)	French → F	1/2	6/12
	Thai=4, wait(T=2, F=2)	Thai → T	2/4	
	Swiss=4, wait(T=2, F=2)	Swiss → F	2/4	
	Italian=2, wait(T=1, F=1)	Italian → T	1/2	
Wait Estimate	0=5, wait(T=4, F=1)	0 → T	1/5	3/12
	20=3, wait(T=2, F=1)	20 → T	1/3	
	40=2, wait(T=1, F=1)	40 → F	1/2	
	60=2, wait(T=0, F=2)	60 → F	0/2	



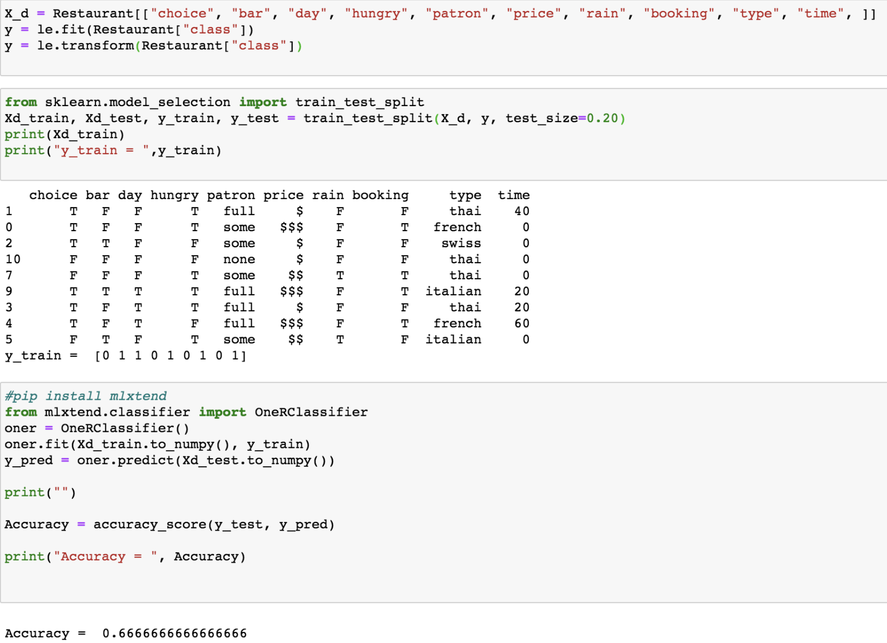
```
import pandas as pd
pd.set_option('display.max_colwidth', None)
Restaurant = pd.read_csv('/Users/catherine/Desktop/NLP/MachineLearning/MachineLearning2021/Restaurant.csv')
print(Restaurant.head(25))
```

```
   choice bar day hungry patron price rain booking     type  time class
0       T   F   F      T   some   $$$    F       T   french     0   yes
1       T   F   F      T   full     $    F       F     thai    40    no
2       T   T   F      F   some     $    F       F    swiss     0   yes
3       T   F   T      T   full     $    F       F     thai    20   yes
4       T   F   T      F   full   $$$    F       T   french    60    no
5       F   T   F      T   some    $$    T       F  italian     0   yes
6       F   T   F      F   none     $    T       F    swiss    20    no
7       F   F   F      T   some    $$    T       T     thai     0   yes
8       F   T   T      F   full     $    T       F    swiss    60    no
9       T   T   T      T   full   $$$    F       T  italian    20    no
10      F   F   F      F   none     $    F       F     thai     0    no
11      T   T   T      T   full     $    F       F    swiss    40   yes
```

```
Restaurant.describe(include='all')
```

	choice	bar	day	hungry	patron	price	rain	booking	type	time	class
count	12	12	12	12	12	12	12	12	12	12.000000	12
unique	2	2	2	2	3	3	2	2	4	NaN	2
top	T	T	F	T	full	$	F	F	thai	NaN	no
freq	7	6	7	7	6	7	8	8	4	NaN	6
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	21.666667	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	23.290003	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.000000	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.000000	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	20.000000	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	40.000000	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	60.000000	NaN

OneRClassifier



```
X_d = Restaurant[["choice", "bar", "day", "hungry", "patron", "price", "rain", "booking", "type", "time", ]]
y = le.fit(Restaurant["class"])
y = le.transform(Restaurant["class"])
```

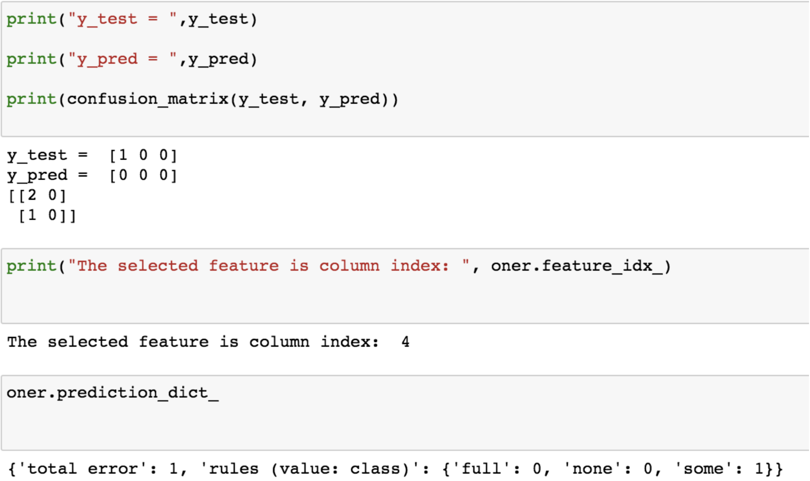
```
from sklearn.model_selection import train_test_split
Xd_train, Xd_test, y_train, y_test = train_test_split(X_d, y, test_size=0.20)
print(Xd_train)
print("y_train = ",y_train)
```

```
   choice bar day hungry patron price rain booking     type  time
1       T   F   F      T   full     $    F       F     thai    40
0       T   F   F      T   some   $$$    F       T   french     0
2       T   T   F      F   some     $    F       F    swiss     0
10      F   F   F      F   none     $    F       F     thai     0
7       F   F   F      T   some    $$    T       T     thai     0
9       T   T   T      T   full   $$$    F       T  italian    20
3       T   F   T      T   full     $    F       F     thai    20
4       T   F   T      F   full   $$$    F       T   french    60
5       F   T   F      T   some    $$    T       F  italian     0
y_train =  [0 1 1 0 1 0 1 0 1]
```

```
#pip install mlxtend
from mlxtend.classifier import OneRClassifier
oner = OneRClassifier()
oner.fit(Xd_train.to_numpy(), y_train)
y_pred = oner.predict(Xd_test.to_numpy())

print("")

Accuracy = accuracy_score(y_test, y_pred)

print("Accuracy = ", Accuracy)
```

```
Accuracy =  0.6666666666666666
```

Selected attribute and rule used



```
print("y_test = ",y_test)

print("y_pred = ",y_pred)

print(confusion_matrix(y_test, y_pred))
```

```
y_test =  [1 0 0]
y_pred =  [0 0 0]
[[2 0]
 [1 0]]
```

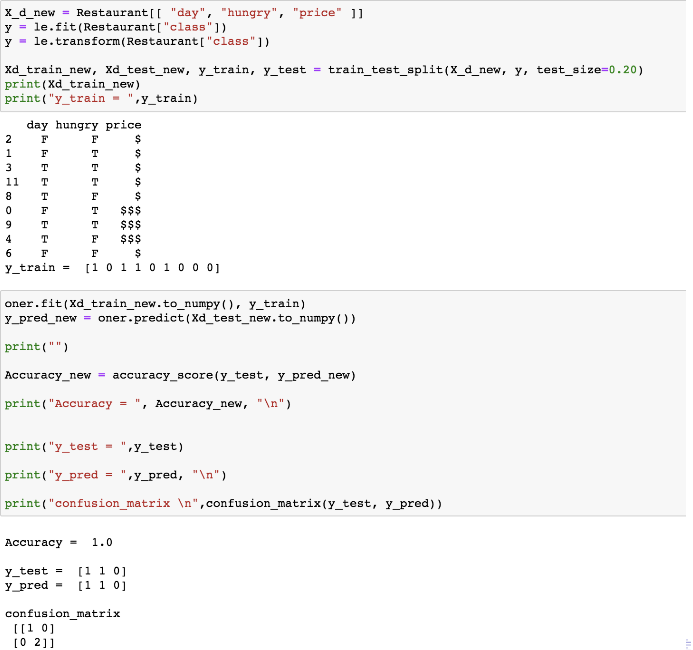
```
print("The selected feature is column index: ", oner.feature_idx_)
```

```
The selected feature is column index:  4
```

```
oner.prediction_dict_
```

```
{'total error': 1, 'rules (value: class)': {'full': 0, 'none': 0, 'some': 1}}
```

Choose only some attributes



```python
X_d_new = Restaurant[[ "day", "hungry", "price" ]]
y = le.fit(Restaurant["class"])
y = le.transform(Restaurant["class"])

Xd_train_new, Xd_test_new, y_train, y_test = train_test_split(X_d_new, y, test_size=0.20)
print(Xd_train_new)
print("y_train = ",y_train)
```

```
   day hungry price
2    F      F     $
1    F      T     $
3    T      T     $
11   T      T     $
8    T      F     $
0    F      T   $$$
9    T      T   $$$
4    T      F   $$$
6    F      F     $
y_train =  [1 0 1 1 0 1 0 0 0]
```

```python
oner.fit(Xd_train_new.to_numpy(), y_train)
y_pred_new = oner.predict(Xd_test_new.to_numpy())

print("")

Accuracy_new = accuracy_score(y_test, y_pred_new)

print("Accuracy = ", Accuracy_new, "\n")


print("y_test = ",y_test)

print("y_pred = ",y_pred, "\n")

print("confusion_matrix \n",confusion_matrix(y_test, y_pred))
```

```
Accuracy =  1.0

y_test =  [1 1 0]
y_pred =  [1 1 0]

confusion_matrix
 [[1 0]
 [0 2]]
```

Series 1 and 2 comments:

- For graphs plotted: check for the patterns at the intersection: is there an increase or decrease of the other graph at the point where they meet? If you plot on the same graph.
- New calculated values should be from one of the technical indicators such as: Daily return, moving average(s), Relative Strength Index (RSI) etc.
- Comment on a calculated value or a generated graph in relation to the question being answered.