Phase 3 Project, book 1

Welcome to my Phase 3 notebook. This notebook will go through my Phase 3 project, which will involve a classification problem. Using logisitc regression, random forests, ensemble methods etc. So let's get started.

The problem:

Using the data below, predict whether or not a bottle of whisky will be "expensive" (>$100USD) or not. I'll be using a number of modeling functions. Potentially this could be used in an app where the user can provide some of the information we'll be using, such as country and type, and it could give a simple yes or no prediction.

Step 1: Importing libraries

Right here I'm importing vitrually every library I can think of that might help with this problem.

In [1]:

```
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import svm
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
import statsmodels as sm
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,preci
sion_score
from sklearn.metrics import recall_score, f1_score,roc_auc_score ,plot_roc_curve, plot_co
nfusion_matrix
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier,AdaBoostClassifier
,GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV


from sklearn.pipeline import Pipeline
```

Step 2: Read the data

In [2]:

```
df=pd.read_csv('Meta-Critic Whisky Database.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	Whisky	Meta Critic	STDEV	#	Cost	Class	Super Cluster	Cluster	Country	Type
0	Macallan 10yo Full Proof 57% 1980 (OB, Giovine...	9.57	0.24	3	$+	SingleMalt-like	ABC	A	Scotland	Malt
1	Ledaig 42yo Dusgadh	9.48	0.23	3	$+	SingleMalt-like	ABC	C	Scotland	Malt
2	Laphroaig 27yo 57.4% 1980-2007 (OB, 5 Oloroso ...	9.42	0.23	4	$+	SingleMalt-like	ABC	C	Scotland	Malt

	Whisky	Meta Critic	STDEV	#	Cost	Class	Super Cluster	Cluster	Country	Type
3	Glenfarclas 40yo	9.29	0.26	17	$+	SingleMalt-like	ABC	A	Scotland	Malt
4	Glengoyne 25yo	9.24	0.22	21	$+	SingleMalt-like	ABC	A	Scotland	Malt

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1767 entries, 0 to 1766
Data columns (total 10 columns):
Whisky           1767 non-null object
Meta Critic      1767 non-null object
STDEV            1767 non-null object
#                1767 non-null int64
Cost             1766 non-null object
Class            1767 non-null object
Super Cluster    1180 non-null object
Cluster          1457 non-null object
Country          1767 non-null object
Type             1767 non-null object
dtypes: int64(1), object(9)
memory usage: 138.2+ KB
```

So there are a few nulls in the data, but not too many. I know I don't want to use 'Super Cluster' in my models, so I can drop that right away.

In [5]:

```
df.drop('Super Cluster',axis=1,inplace=True)
```

Step 3: Assign the target variable

I know that Cost is going to be my target variable. So let's look at it real quick.

In [6]:

```
df['Cost'].value_counts()
```

Out[6]:

```
$$$$      601
$$$$$     352
$$$       334
$$        205
$$$$$+    185
$          89
Name: Cost, dtype: int64
```

And we know there's 1 null value.

In [7]:

```
df.loc[df.Cost.isna()]
```

Out[7]:

	Whisky	Meta Critic	STDEV	#	Cost	Class	Cluster	Country	Type
587	Teaninich 10yo (F&F)	8.29	0.18	4	NaN	SingleMalt-like	F	Scotland	Malt

What I'd like to do is get these $'s into numbers, then into 1's and 0's. Let's try to see if we can get a length for these elements. First things first, let's get rid of that single null value.

Step 4: Process the data

Step 4: Process the data

In [8]:

```
df=df.dropna(subset=['Cost'])
```

And from further exploration of the data, I know there's 1 invalid value in 'Meta Critic', so let's look at that and get rid of it.

In [9]:

```
df.loc[df['Meta Critic'] == '#REF!']
```

Out[9]:

	Whisky	Meta Critic	STDEV	#	Cost	Class	Cluster	Country	Type
1179	Bruichladdich Octomore 10 (Fourth Edition)	#REF!	#REF!	3	$	SingleMalt-like	J	Scotland	Malt

In [10]:

```
i=df[((df['Meta Critic'] == '#REF!') )].index
```

In [11]:

```
df.drop(i, inplace=True)
```

Getting back to our target variable. I think the easiest way to do this is to do my sorting by length.

In [12]:

```
len(df['Cost'][0])
```

Out[12]:

```
6
```

In [13]:

```
len(df['Cost'][55])
```

Out[13]:

```
5
```

So what've learned is we can split these up by length. And we know that 4 dollar signs is about $100 USD, that's a nice place to split it.

In [14]:

```
Target=df['Cost']
df1=df.drop('Cost',axis=1)
```

In [15]:

```
df1.head()
```

Out[15]:

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type
0	Macallan 10yo Full Proof 57% 1980 (OB, Giovine...	9.57	0.24	3	SingleMalt-like	A	Scotland	Malt
1	Ledaig 42yo Dusgadh	9.48	0.23	3	SingleMalt-like	C	Scotland	Malt
2	Laphroaig 27yo 57.4% 1980-2007 (OB, 5 Oloroso ...	9.42	0.23	4	SingleMalt-like	C	Scotland	Malt
3	Glenfarclas 40yo	9.29	0.26	17	SingleMalt-like	A	Scotland	Malt
4	Glengoyne 25yo	9.24	0.22	21	SingleMalt-like	A	Scotland	Malt

Ok, so we got our target variable separate from the data. Now we need to do a few things to get the data in working condition. First we need our target dataset to be a 1 or 0 based on our criteria. So anything above 4 dollar signs will be classified as 'expensive', it'll get assigned to 1. Everything else will be 0. So let's make that happen.

In [16]:

```
type(Target)
```

Out[16]:

```
pandas.core.series.Series
```

In [17]:

```
Target.unique()
```

Out[17]:

```
array(['$$$$$+', '$$$$$', '$$$$', '$$', '$$$', '$'], dtype=object)
```

In [18]:

```
len(Target[0])
```

Out[18]:

```
6
```

Here's where we're going to sort our 'Cost' set:

In [19]:

```
y=[]
for i in Target.index:
    if len(Target[i]) <=4:
        y.append(0)
    else:
        y.append(1)
```

In [20]:

```
y=pd.Series(y)
```

In [21]:

```
y.unique()
```

Out[21]:

```
array([1, 0], dtype=int64)
```

In [22]:

```
y.value_counts(normalize=True)
```

Out[22]:

```
0    0.696317
1    0.303683
dtype: float64
```

About a 70/30 split. It's a little skewed, but hopefully not too much.

In [23]:

```
y.head(15)
```

Out[23]:

```
0     1
1     1
2     1
3     1
4     1
5     1
6     1
7     1
8     1
9     1
10    0
11    0
12    1
13    1
14    1
dtype: int64
```

In [24]:

```
len(y)
```

Out[24]:

```
1765
```

Ok, we have us a target dataset. Now I think I should do a little more to the data before we do the train test split

In [25]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1765 entries, 0 to 1766
Data columns (total 8 columns):
Whisky         1765 non-null object
Meta Critic    1765 non-null object
STDEV          1765 non-null object
#              1765 non-null int64
Class          1765 non-null object
Cluster        1455 non-null object
Country        1765 non-null object
Type           1765 non-null object
dtypes: int64(1), object(7)
memory usage: 204.1+ KB
```

So most of these names have an age in them. I'm going to try to get that age out and make it into a column, that should be significant in the price of the whisky.

In [26]:

```
df['Whisky'][0].split()
```

Out[26]:

```
['Macallan',
 '10yo',
 'Full',
 'Proof',
 '57%',
 '1980',
 '(OB,',
 'Giovinetti',
 '&',
 'Figli)']
```

In [27]:

```
df['Whisky'][0].split()[1]
```

Out[27]:

'10yo'

In [28]:

```
df['Whisky'][0].split()[1].endswith('yo')
```

Out[28]:

```
True
```

In [29]:

```
test_age=df['Whisky'][0].split()[1][:2]
```

In [30]:

```
test_age
```

Out[30]:

```
'10'
```

In [31]:

```
Age=[]
for a in range(0, len(df['Whisky'][0].split())):
    split_name=df['Whisky'][0].split()[a]
    if split_name.endswith('yo') == True:
        y_string=df['Whisky'][0].split()[a][:2] #I'm assuming none of these bottles is o
ver 100 years old
        Age.append(y_string)
```

In [32]:

```
Age
```

Out[32]:

```
['10']
```

Ok, so we've successfully separated the age from the name. So let's make another loop, that goes through all the whiskies. I've added another bit to the 'if' statement, since there's a whisky that ends with 'yo', so I need to make sure the characters before the 'yo' are numbers.

In [33]:

```
#I ended up going another route with making the ages, but this works, so I'd like to keep
it in.
import re
Ages=[]
for a in df['Whisky'].index:
    whisky_split=df['Whisky'][a].split()
    tracker=0
    for b in range(0, len(whisky_split)):
        sing_word=whisky_split[b]
        if sing_word.endswith('yo') == True and any(char.isdigit() for char in sing_word
):
            tracker=1
            y_string= ''.join(c for c in sing_word if c.isdigit()) #re.sub("[^0-9]", "",
sing_word) is another way to do this
            Ages.append(y_string)
    if tracker==0:
        Ages.append('Unknown')
```

In [34]:

```
df2=df1.copy()
```

In [35]:

```
df2['Ages']='Unknown'
```

In [36]:

```
df2.head()
```

Out[36]:

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages
0	Macallan 10yo Full Proof 57% 1980 (OB, Giovine...	9.57	0.24	3	SingleMalt-like	A	Scotland	Malt	Unknown
1	Ledaig 42yo Dusgadh	9.48	0.23	3	SingleMalt-like	C	Scotland	Malt	Unknown
2	Laphroaig 27yo 57.4% 1980-2007 (OB, 5 Oloroso ...	9.42	0.23	4	SingleMalt-like	C	Scotland	Malt	Unknown
3	Glenfarclas 40yo	9.29	0.26	17	SingleMalt-like	A	Scotland	Malt	Unknown
4	Glengoyne 25yo	9.24	0.22	21	SingleMalt-like	A	Scotland	Malt	Unknown

In [37]:

```
for i in df2.index:
    split_name=df2['Whisky'][i].split()
    for item in split_name  :
        if item.endswith('yo') and any(char.isdigit() for char in item):
            df2.at[i,'Ages']=item[:-2]
```

In [38]:

```
df2.head()
```

Out[38]:

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages
0	Macallan 10yo Full Proof 57% 1980 (OB, Giovine...	9.57	0.24	3	SingleMalt-like	A	Scotland	Malt	10
1	Ledaig 42yo Dusgadh	9.48	0.23	3	SingleMalt-like	C	Scotland	Malt	42
2	Laphroaig 27yo 57.4% 1980-2007 (OB, 5 Oloroso ...	9.42	0.23	4	SingleMalt-like	C	Scotland	Malt	27
3	Glenfarclas 40yo	9.29	0.26	17	SingleMalt-like	A	Scotland	Malt	40
4	Glengoyne 25yo	9.24	0.22	21	SingleMalt-like	A	Scotland	Malt	25

In [39]:

```
df2['Ages']
```

Out[39]:

```
0            10
1            42
2            27
3            40
4            25
          ...
1762    Unknown
1763    Unknown
1764    Unknown
1765    Unknown
1766    Unknown
Name: Ages, Length: 1765, dtype: object
```

So we're going to write a few lines of code and find all the years that aren't numbers, or aren't 'Unknown'.

In [40]:

```
df3 = df2[~df2["Ages"].str.isdigit()]
```

In [41]:

```
df3
```

Out[41]:

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages
5	Amrut Spectrum (Batch 1)	9.21	0.24	14	SingleMalt-like	C	India	Malt	Unknown
9	Balvenie TUN 1401 (all batches)	9.15	0.31	19	SingleMalt-like	A	Scotland	Malt	Unknown
10	Aberlour A'Bunadh (Batch 33)	9.14	0.14	5	SingleMalt-like	A	Scotland	Malt	Unknown
11	Aberlour A'Bunadh (Batch 37)	9.13	0.10	3	SingleMalt-like	A	Scotland	Malt	Unknown
16	Aberlour A'Bunadh (Batch 40)	9.11	0.15	4	SingleMalt-like	A	Scotland	Malt	Unknown
...	...	...	...	...	...	...	...	...	...
1762	Jim Beam White Label	7.64	0.56	22	Bourbon-like	R2	USA	Bourbon	Unknown
1763	Rebel Yell Kentucky Bourbon	7.56	0.67	14	Bourbon-like	R0	USA	Bourbon	Unknown
1764	Jim Beam Red Stag (Black Cherry)	7.35	1.01	4	Bourbon-like	NaN	USA	Flavoured	Unknown
1765	Virginia Black	7.19	1.23	6	Bourbon-like	R2	USA	Bourbon	Unknown
1766	Early Times Kentucky Whisky	7.10	0.49	7	Bourbon-like	R1	USA	Bourbon	Unknown

1117 rows × 9 columns

In [42]:

```
df3a=df3[~df3['Ages'].str.contains("Unknown")]
```

In [43]:

```
df3a
```

Out[43]:

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages
23	Amrut Greedy Angels (8yo and 10yo)	9.07	0.20	13	SingleMalt-like	C	India	Malt	(8

When there are 2 ages on a bottle, always take the younger number.

In [44]:

```
df2.at[23, 'Ages']='8'
```

In [45]:

```
print(df2.loc[23])
```

```
Whisky          Amrut Greedy Angels (8yo and 10yo)
Meta Critic                                   9.07
STDEV                                         0.20
#                                               13
Class                              SingleMalt-like
Cluster                                          C
Country                                      India
Type                                          Malt
Ages                                             8
Name: 23, dtype: object
```

In [46]:

```
df2['Ages'].describe()
```

Out[46]:

```
count        1765
unique         37
top       Unknown
freq         1116
Name: Ages, dtype: object
```

That looks good so far. Even the right length.

So the majority of them are either unknowns, or don't follow the convention.

In [47]:

```
df2.head(40)
```

Out[47]:

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages
0	Macallan 10yo Full Proof 57% 1980 (OB, Giovine...	9.57	0.24	3	SingleMalt-like	A	Scotland	Malt	10
1	Ledaig 42yo Dusgadh	9.48	0.23	3	SingleMalt-like	C	Scotland	Malt	42
2	Laphroaig 27yo 57.4% 1980-2007 (OB, 5 Oloroso ...	9.42	0.23	4	SingleMalt-like	C	Scotland	Malt	27
3	Glenfarclas 40yo	9.29	0.26	17	SingleMalt-like	A	Scotland	Malt	40
4	Glengoyne 25yo	9.24	0.22	21	SingleMalt-like	A	Scotland	Malt	25
5	Amrut Spectrum (Batch 1)	9.21	0.24	14	SingleMalt-like	C	India	Malt	Unknown
6	Highland Park 25yo	9.18	0.19	18	SingleMalt-like	C	Scotland	Malt	25
7	Highland Park 40yo	9.16	0.43	10	SingleMalt-like	C	Scotland	Malt	40
8	Tamdhu 30yo (MacPhail Collection 2009)	9.16	0.18	3	SingleMalt-like	A	Scotland	Malt	30
9	Balvenie TUN 1401 (all batches)	9.15	0.31	19	SingleMalt-like	A	Scotland	Malt	Unknown
10	Aberlour A'Bunadh (Batch 33)	9.14	0.14	5	SingleMalt-like	A	Scotland	Malt	Unknown
11	Aberlour A'Bunadh (Batch 37)	9.13	0.10	3	SingleMalt-like	A	Scotland	Malt	Unknown
12	Highland Park 30yo	9.13	0.41	14	SingleMalt-like	C	Scotland	Malt	30
13	Laphroaig 25yo	9.13	0.28	23	SingleMalt-like	C	Scotland	Malt	25
14	Nikka Single Cask Coffey Malt 12yo	9.12	0.48	9	SingleMalt-like	C	Japan	Malt	12
15	Yamazaki 18yo	9.12	0.27	23	SingleMalt-like	C	Japan	Malt	18
16	Aberlour A'Bunadh (Batch 40)	9.11	0.15	4	SingleMalt-like	A	Scotland	Malt	Unknown
17	Aberlour A'Bunadh (Batch 49)	9.10	0.20	10	SingleMalt-like	A	Scotland	Malt	Unknown
18	Aberlour A'Bunadh (Batch 56)	9.09	0.22	5	SingleMalt-like	A	Scotland	Malt	Unknown
19	Compass Box Last Vatted Malt	9.09	0.16	4	SingleMalt-like	C	Scotland	Malt	Unknown

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages
20	Redbreast 21yo	9.09	0.29	20	SingleMalt-like	C	Ireland	Malt	21
21	Yamazaki Sherry Cask (all vintages)	9.09	0.33	11	SingleMalt-like	A	Japan	Malt	Unknown
22	Bruichladdich 21yo Cuvée 407 PX	9.08	0.17	8	SingleMalt-like	A	Scotland	Malt	21
23	Amrut Greedy Angels (8yo and 10yo)	9.07	0.20	13	SingleMalt-like	C	India	Malt	8
24	Amrut Spectrum (all batches)	9.07	0.38	18	SingleMalt-like	C	India	Malt	Unknown
25	BenRiach 18yo Albariza Pedro Ximenez Peated	9.07	0.19	8	SingleMalt-like	C	Scotland	Malt	18
26	Highland Park 16yo Odin	9.07	0.28	7	SingleMalt-like	C	Scotland	Malt	16
27	Bowmore Springtide	9.06	0.67	5	SingleMalt-like	C	Scotland	Malt	Unknown
28	GlenDronach Cask Strength (batch 1)	9.05	0.27	6	SingleMalt-like	A	Scotland	Malt	Unknown
29	Kavalan Solist Vinho Barrique	9.05	0.21	20	SingleMalt-like	A	Taiwan	Malt	Unknown
30	Macallan Cask Strength	9.05	0.38	22	SingleMalt-like	A	Scotland	Malt	Unknown
31	Aberlour A'Bunadh (Batch 53)	9.04	0.12	6	SingleMalt-like	A	Scotland	Malt	Unknown
32	GlenDronach Cask Strength (batch 2)	9.04	0.14	10	SingleMalt-like	A	Scotland	Malt	Unknown
33	Highland Park 18yo	9.04	0.24	32	SingleMalt-like	C	Scotland	Malt	18
34	Kavalan Solist PX Cask	9.04	0.51	8	SingleMalt-like	A	Taiwan	Malt	Unknown
35	Sheep Dip Old Hebridean 1990 Blended Malt	9.04	0.22	7	SingleMalt-like	C	Scotland	Malt	Unknown
36	Aberlour A'Bunadh (Batch 39)	9.03	0.22	8	SingleMalt-like	A	Scotland	Malt	Unknown
37	Highland Park 17yo The Dark	9.03	0.38	13	SingleMalt-like	C	Scotland	Malt	17
38	Karuizawa 1990 Sherry Butt	9.03	0.30	4	SingleMalt-like	A	Japan	Malt	Unknown
39	Timorous Beastie 21yo Sherry Edition	9.03	0.2	3	SingleMalt-like	B	Scotland	Malt	21

In [48]:

```
Unk_ages=df2.loc[df2['Ages']=='Unknown']
```

In [49]:

```
uniqs=list(Unk_ages['Whisky'].unique())
```

In [50]:

```
uniqs
```

Out[50]:

```
['Amrut Spectrum (Batch 1)',
 'Balvenie TUN 1401 (all batches)',
 "Aberlour A'Bunadh (Batch 33)",
 "Aberlour A'Bunadh (Batch 37)",
 "Aberlour A'Bunadh (Batch 40)",
 "Aberlour A'Bunadh (Batch 49)",
```

```
    "Aberlour A'Bunadh (Batch 56)",
    'Compass Box Last Vatted Malt',
    'Yamazaki Sherry Cask (all vintages)',
    'Amrut Spectrum (all batches)',
    'Bowmore Springtide',
    'GlenDronach Cask Strength (batch 1)',
    'Kavalan Solist Vinho Barrique',
    'Macallan Cask Strength',
    "Aberlour A'Bunadh (Batch 53)",
    'GlenDronach Cask Strength (batch 2)',
    'Kavalan Solist PX Cask',
    'Sheep Dip Old Hebridean 1990 Blended Malt',
    "Aberlour A'Bunadh (Batch 39)",
    'Karuizawa 1990 Sherry Butt',
    'Amrut Spectrum 004 (Batch 2)',
    'Kavalan Solist Manzanilla Cask',
    "Aberlour A'Bunadh (Batch 44)",
    "Aberlour A'Bunadh (Batch 52)",
    'Kavalan Solist Amontillado Cask',
    "Aberlour A'Bunadh (Batch 35)",
    'Arran Malt 21st Anniversary Edition',
    'Laphroaig Cairdeas 2013 Port Wood',
    "Aberlour A'Bunadh (Batch 45)",
    "Aberlour A'Bunadh (Batch 36)",
    'Amrut PX Sherry Single Cask 2696 (LCBO)',
    'Kilchoman Port Cask Matured',
    "Aberlour A'Bunadh (Batch 42)",
    'Amrut Portonova',
    'Glenlivet Alpha',
    'Glenmorangie Signet',
    "Aberlour A'Bunadh (Batch 30)",
    'AnCnoc 1975',
    'Hakushu Sherry Cask',
    'Kavalan Solist Sherry Cask',
    "Aberlour A'Bunadh (Batch 58)",
    'Kavalan Solist Port Cask',
    "Aberlour A'Bunadh (Batch 34)",
    'Amrut Madeira Cask Finish (Batch 1)',
    "Aberlour A'Bunadh (all batches)",
    'Balvenie TUN 1509 (all batches)',
    'GlenDronach Cask Strength (batch 4)',
    "Aberlour A'Bunadh (Batch 54)",
    'Amrut Intermediate Sherry',
    'Amrut Kadhambam',
    "Aberlour A'Bunadh (Batch 46)",
    'Glengoyne Cask Strength (batch 7)',
    'Kilchoman PX Sherry Finish',
    'Amrut PX Sherry Single Cask 3516 (SAQ)',
    'Compass Box This is Not a Luxury Whisky',
    'GlenDronach Cask Strength (all batches)',
    'GlenDronach Cask Strength (batch 3)',
    "Aberlour A'Bunadh (Batch 47)",
    'Glen Garioch 1998 Wine Cask Matured',
    'Kilchoman Loch Gorm',
    'Smogen Sherry Project 1:2',
    "Aberlour A'Bunadh (Batch 66)",
    'Balblair 1990 (all releases)',
    'GlenDronach Cask Strength (batch 6)',
    'High Coast (Box) PX - Pedro Ximénez Finish',
    'Kavalan Solist Fino Sherry Cask',
    "Aberlour A'Bunadh (Batch 32)",
    "Aberlour A'Bunadh (Batch 61)",
    'Amrut PX Sherry Single Cask (all casks)',
    'Glenmorangie Companta',
    'Green Spot Chateau Leoville Barton',
    "Aberlour A'Bunadh (Batch 59)",
    "Aberlour A'Bunadh (Batch 60)",
    'Glenlivet Nadurra Cask Strength (NAS)',
    'High Coast (Box) Dalvve Sherry Influence',
    'Smogen Sherry Project 1:4',
    'Kavalan Brandy Oak',
    'Bruichladdich Port Charlotte 2009 MC:01',
```

```
    'Deanston 2008 Bordeaux Red Wine Cask Matured',
    'Glenfiddich Snow Phoenix',
    'Glengoyne Teapot Dram (all batches)',
    'Glenmorangie Bacalta',
    "Glenrothes Minister's Reserve",
    'Highland Park Full Volume',
    "Aberlour A'Bunadh (Batch 48)",
    "Aberlour A'Bunadh (Batch 65)",
    'Arran Malt Amarone Cask Finish',
    'Macallan Reflexion',
    'Two Brewers Release 09 Special Finishes',
    "Aberlour A'Bunadh (Batch 50)",
    'Smogen Sherry Project 1:3',
    'Arran Malt Sassicaia Wine Cask Finish',
    'Macallan Classic Cut (all editions)',
    'Glenfarclas 105',
    'Macallan Ruby',
    'Two Brewers Special Finishes (all releases)',
    'GlenDronach Cask Strength (batch 5)',
    'Glengoyne Cask Strength (batch 1)',
    'Highland Park Valknut',
    'Scallywag Cask Strength (all batches)',
    "Aberlour A'Bunadh (Batch 57)",
    'GlenDronach Cask Strength (batch 7)',
    "Glenmorangie Nectar d'Or",
    "Glenrothes Whisky Maker's Cut",
    'Karuizawa Asama Vintages 1999-2000',
    'Sheep Dip 1999 Amoroso Blended Malt',
    'Smogen Sherry Project 1:1',
    'Two Brewers Release 15 Special Finishes',
    'Jura Brooklyn',
    'Macallan Rare Cask (all batches)',
    'Macallan Sienna',
    'Teerenpeli Distiller’s Choice KARHI',
    'Amrut Bengal Tiger PX Single Cask (Canada)',
    'Two Brewers Release 02 Special Finishes',
    "Aberlour A'Bunadh (Batch 38)",
    "Aberlour A'Bunadh (Batch 62)",
    'Redbreast Lustau Edition',
    'Tomatin Cu Bocan 2005 Limited Edition',
    'Arran Malt Madeira Wine Cask',
    'Glengoyne Cask Strength (all batches)',
    'Glengoyne Cask Strength (batch 2)',
    'Glenmorangie The Taghta',
    'Oban Distillers Edition (all vintages)',
    'Compass Box The Story of the Spaniard',
    'Ohishi Sherry Single Cask',
    'Redbreast Mano a Lámh',
    'Sullivans Cove French Oak',
    'Ardbeg Galileo',
    'Balblair 1989',
    'Bunnahabhain Moine (all bottlings)',
    'Tamdhu Batch Strength (all batches)',
    'Kilchoman Sherry Single Cask',
    'AnCnoc 2000',
    'Arran Malt Port Cask Finish',
    'Highland Park Valkyrie',
    'Swiss Highland Classic Single Malt',
    'Highland Park Magnus (2017)',
    'Dalmore Port Wood Reserve',
    'Glenmorangie Dornoch',
    'Glenmorangie Quinta Ruban',
    'Kilchoman Madeira Cask Matured',
    'Amrut Naarangi',
    'Laphroaig PX Triple Matured',
    'Santis Alpstein (all editions)',
    'Spirit of Hven Sankt Claus',
    'Glen Garioch 1999 Sherry Cask Matured',
    'Glenfarclas 1968-2000 54.2% (OB, Old Stock Reserve, Ceramic)',
    'Glenmorangie A Midwinter Night’s Dram',
    'Dalwhinnie Distillers Edition',
    'Aberlour Casg Annamh',
```

```
    'Talisker Port Ruighe',
    'Tomatin Oloroso Sherry 1995',
    'Glengoyne Cask Strength (batch 4)',
    'Glenmorangie Duthac',
    "Macallan Whisky Maker's Edition",
    'Redbreast All Sherry Single Cask 1999',
    'Teerenpeli Distiller’s Choice AURA',
    'Amrut PX Sherry Single Cask 2701',
    'Kavalan Sherry Oak',
    'Mackmyra Skordetid',
    'Arran Malt Sherry Single Cask',
    'Glenlivet Nadurra Oloroso',
    'Mackmyra Midvinter',
    'Penderyn Portwood',
    'Westland Winter 2016',
    'Glengoyne Cask Strength (batch 3)',
    'Highland Park Dark Origins',
    'Ledaig 1996',
    'Smogen Primör',
    'Westland Sherry Wood',
    'Glenmorangie Milsean',
    'Green Spot Chateau Montelena',
    'Dalmore Cigar Malt',
    'Glenmorangie The Tayne',
    'Glenmorangie Lasanta',
    'Bruichladdich Rocks',
    'Glen Moray Classic Sherry Cask finish',
    'Arran Malt Sherry Cask Finish',
    'Wemyss Malts The Spice King',
    'Bushmills Black Bush',
    'Glen Moray Classic Port Cask finish',
    'Teeling Single Malt',
    'Arran Malt Pomerol Bordeaux Cask Finish',
    'Kavalan Concertmaster Port Cask',
    'Timorous Beastie',
    'Glen Scotia Double Cask',
    "Glenfiddich Malt Master's Edition",
    'Laphroaig Select',
    'Tomatin Cu Bocan Sherry Edition',
    'Auchentoshan Three Wood',
    'Dalmore Cigar Malt Reserve',
    'Glenfiddich Reserve Cask',
    'Dalmore King Alexander III',
    'Jura Seven Wood',
    'Ohishi Sherry Cask',
    'Writers Tears Red Head Single Malt',
    'Game of Thrones House Baratheon Royal Lochnagar 12 ans',
    'Bushmills Sherry Cask Reserve',
    'Singleton of Dufftown Tailfire',
    'Bowmore Black Rock',
    'Glenrothes Robur Reserve',
    'Macallan Select Oak',
    'Wemyss Malts Velvet Fig',
    'Glenglassaugh Revival',
    'Nantou (Omar) Yushan Sherry Cask',
    'Tullibardine 500 Sherry Finish',
    'Mackmyra Blomstertid',
    'Glenrothes Sherry Cask Reserve',
    'Dalmore Gran Reserva',
    'Bladnoch Samsara',
    "Longmorn Distiller's Choice",
    'Penderyn Madeira',
    'Lohin McKinnon Wine Barrel Finished (Black Sage)',
    'Tullibardine 1993 Port',
    'Penderyn Sherrywood',
    'Jura Turas Mara',
    'Amrut PX Sherry Single Cask 2702',
    'Singleton of Dufftown Spey Cascade',
    "Ichiro's Malt The Joker",
    'Bruichladdich Black Art 7.x 1994',
    'Bruichladdich Black Art 6.x 1990',
    'Bruichladdich Black Art 3.x 1989',
```

```
'Midleton Very Rare 2017',
'Midleton Dair Ghaelach',
'Amrut Virgin Oak Single Cask',
'Bruichladdich Black Art 2.x 1989',
'Mackmyra Reserve Single Cask (various casks)',
'Amrut Double Cask',
'Yamazaki Mizunara',
'Bruichladdich The Organic 2010',
'Kanosuke New Born 2018 8mo',
'Glenlivet Nadurra First Fill (White Oak)',
'Yamazaki Limited Edition 2016',
'Tomatin Decades',
'Amrut Herald',
'Bruichladdich Black Art 5.x 1992',
'Amrut Fusion',
'Glenmorangie Ealanta',
'Kavalan ex-Bourbon Oak',
'Macallan Edition No. 2',
'Compass Box Spice Tree Extravaganza',
'Macallan Edition No. 4',
'Arran Malt The Devil’s Punch Bowl (all chapters)',
'Arran Malt Bourbon Single Cask',
'Compass Box Phenomenology',
'Macallan Edition No. 6',
'Glenmorangie Astar',
'Kavalan Solist Ex-Bourbon',
'Mackmyra Moment Glöd',
'Mackmyra Moment Urberg',
'Midleton Very Rare (all vintages)',
'Glen Grant Five Decades',
'High Coast (Box) The 2nd Step Collection 02',
'Paul John Single Cask',
'Amrut Two Continents',
'Game of Thrones House Tyrell Clynelish Reserve',
'Midleton Very Rare 2016',
'Nikka From the Barrel',
'Macallan Edition No. 1',
'Arran Malt Tokaji Aszu Wine Finish',
'Two Brewers Release 14 Innovative',
'Two Brewers Release 21 Classic',
'Westland Single Cask',
'Compass Box Spice Tree',
'Mortlach Special Strength',
'Cragganmore NAS (Special Release 2016)',
'Macallan Edition No. 3',
'Arran Malt Napoleon Cognac Finish',
'Glenfiddich Cask of Dreams',
'Mackmyra Special 04',
'Amrut Bourbon Single Cask',
'Mackmyra Moment Rimfrost',
'Bruichladdich Black Art 4.x 1990',
'Glenfiddich Project XX Experimental Series No. 2',
'High Coast (Box) The Festival 2016',
'Shelter Point French Oak Double Barreled',
'Macallan Edition No. 5',
'Mackmyra Special 03',
'Bruichladdich The Organic 2009',
"Ichiro's Malt Double Distilleries",
'Nikka Coffey Malt',
'Two Brewers Release 17 Innovative',
'Two Brewers Release 04 Special Finishes',
'Bruichladdich The Organic (Mid Coul, Coulmore, Mains of Tullibardine Farms)',
'Matsui Sakura Cask',
"Aberlour A'Bunadh Alba (all batches)",
"Aberlour A'Bunadh Alba (Batch 1)",
'Balcones Texas Single Malt Whisky',
"Teerenpeli Distiller's Choice KASKI",
'Kilkerran Work in Progress Sherry Wood',
'Sullivans Cove Port Cask Strength',
'Glenrothes Vintage 1995 (all bottlings)',
'Bruichladdich The Organic (all editions)',
'Glen Scotia Victoriana',
```

```
'Glenlivet Cipher',
'Mackmyra Special 09',
'Paul John Classic Select Cask',
'Compass Box Oak Cross',
'Sullivans Cove Bourbon Cask Strength',
'Two Brewers Innovative (all releases)',
'Midleton Very Rare 2015',
'Sullivans Cove American Oak',
'Arran Malt Sauternes Finish',
'Nikka Pure Malt Red',
'Westland American Single Malt (American Oak)',
"Auchentoshan Bartender's Malt",
'Benromach Organic',
'Green Spot',
'High Coast (Box) The Messenger',
'Copperworks American Single Malt',
'Two Brewers Release 08 Innovative',
'Teerenpeli Distiller’s Choice RASI',
"Yamazaki Distiller's Reserve",
'Mackmyra Moment Jord',
'Mackmyra Special 05',
'Bruichladdich The Organic 2003',
'Mackmyra Moment Källa',
'Paul John Brilliance',
'Mackmyra Moment Solsken',
'Bunnahabhain Stiuireadair',
'Mortlach Rare Old',
'Mackmyra Special 08',
'Glenfiddich IPA Cask Finish Experimental Series No. 1',
'Bunnahabhain Darach Ur',
'Bunnahabhain Eirigh Na Greine',
"Glenlivet Captain's Reserve",
'Mackmyra Moment Malström (Maelstrom)',
'Nikka All Malt',
'Mackmyra Special 10',
'Okanagan Spirits Laird of Fintry (all editions)',
'Glenrothes Vintage 1998 (2014)',
'Glenrothes Vintage Reserve (NAS)',
'Old Pulteney Navigator',
'Gouden Carolus Single Malt',
'Monkey Shoulder',
'Sullivans Cove Double Cask',
'Teeling Small Batch (Rum Cask Finish)',
'Tullibardine 225 Sauternes Finish',
'Nantou (Omar) Yushan Bourbon Cask',
'Wemyss Malts Smooth Gentleman',
'Bruichladdich Black Art 1989',
'Macallan Amber',
"Glenrothes Elders' Reserve",
'Stalk & Barrel Single Malt (All Casks)',
'Scallywag',
'Glencadam Origin 1825 (NAS)',
'Lohin McKinnon Choclolate Malt',
'Bruichladdich Sherry Classic',
'Mackmyra Moment Jakt',
'Glenfiddich Select Cask',
'The Irishman Single Malt (NAS)',
'Arran Malt Robert Burns Single Malt',
'Tullibardine 1993 Sauternes',
'Glenrothes Manse Reserve',
'Glenrothes Vintage 2001 (all bottlings)',
'Ohishi Brandy Cask',
'Glen Garioch Virgin Oak',
'Two Brewers Release 05 Innovative',
'Singleton of Dufftown Unité',
'Tullibardine 228 Burgundy Finish',
'Dalmore Valour',
"Glenlivet Founder's Reserve",
'Santis Edition Sigel',
'Scapa Skiren',
'Game of Thrones House Tully Singleton Glendullan Select',
'Penderyn Legend',
```

```
    'Singleton of Glen Ord Signature',
    'Glenrothes Select Reserve',
    'Penderyn Aur Cymru',
    'Mars Iwai Tradition',
    'Mackmyra Special 01',
    'Santis Edition Säntis',
    "McClelland's Speyside Single Malt",
    'Kavalan Solist Moscatel Cask',
    'Midleton Barry Crockett Legacy',
    'Yamazaki Limited Edition 2015',
    'Mackmyra Iskristall',
    'High Coast (Box) The 2nd Step Collection 03',
    "Mackinlay's Shackleton Rare Old Highland Malt Discovery edition",
    'Compass Box Double Single',
    "Ichiro's Malt Chichibu On The Way",
    'Mackmyra Vinterdrom',
    'Compass Box Double Single (all editions)',
    'Kavalan Distillery Reserve Rum Cask',
    'High Coast (Box) Quercus IV Mongolica',
    'Arran Malt Orkney Bere Barley',
    "Mackinlay's Shackleton Rare Old Highland Malt (both limited editions)",
    'Matsui Mizunara Cask',
    'Amrut Cask Strength',
    'Yamazaki Bourbon Barrel',
    'Two Brewers Cask Strength (all releases)',
    'Two Brewers Release 10 Cask Strength',
    'Glenmorangie Spios Private Edition No 9',
    'Glenmorangie Tusail',
    'Kilkerran Work in Progress Bourbon Wood',
    'Compass Box Enlightenment',
    'Two Brewers Release 06 Classic',
    'Compass Box Rivals',
    'BenRiach Cask Strength',
    'Kavalan Podium',
    'Mackmyra The First Edition (Den Första Utgåvan)',
    'Two Brewers Release 13 Classic',
    'Glenmorangie The Tarlogan',
    'Two Brewers Classic (all releases)',
    "Mackinlay's Shackleton Rare Old Highland Malt Journey edition",
    'Kilkerran Work in Progress',
    'Westland Garryana',
    'Mackmyra Preludium 03',
    'Bruichladdich Classic Laddie Scottish Barley',
    'Mars Maltage Cosmo',
    'Yamazaki Puncheon',
    'Spirit of Hven Urania',
    'Two Brewers Release 16 Classic',
    'Two Brewers Release 01 Classic',
    "Ichiro's Malt Chichibu The Floor Malted",
    'Nikka Miyagikyo NAS',
    'Tomatin Cu Bocan Virgin Oak Edition',
    'AnCnoc Peter Arkle (all releases)',
    'Mackmyra Svensk Ek',
    'Old Pulteney Huddart',
    "Ichiro's Malt Chichibu The First",
    'Kavalan King Car Conductor',
    'Balblair 2000',
    'High Coast (Box) Quercus III Petraea',
    'Mackmyra Special 07',
    'Yamazaki NAS',
    'Sheep Dip Blended Malt',
    'Benromach Traditional',
    'Game of Thrones House Stark Dalwhinnie Winter’s Frost',
    "Glen Garioch Founder's Reserve",
    'Bruichladdich Laddie Classic (Edition 01)',
    'FEW Single Malt',
    'Kavalan Single Malt Whisky',
    'Talisker Skye',
    'Nikka Taketsuru NAS',
    'Mackmyra The Swedish Whisky (Brukswhisky)',
    'Wemyss Malts The Hive',
    "Hakushu Distiller's Reserve",
```

```
    'Hibiki Harmony',
    'Auchentoshan Valinch',
    "Ichiro's Malt Mizunara Wood Reserve (MWR)",
    'Tomatin Cask Strength',
    'Hakushu NAS',
    "Glenkinchie Distiller's Edition (all editions)",
    'Glenglassaugh Octaves Classic',
    'Amrut Indian Single Malt',
    'Mackmyra Special 02',
    'Penderyn Myth',
    'Wolfburn (NAS)',
    "Mackinlay's Shackleton Blended Malt",
    'Arran Malt Lochranza Reserve',
    'Glenfiddich 1963 Original Malt',
    "Hibiki Harmony Master's Select",
    'High Coast (Box) Quercus I Robur',
    'Highland Park Svein',
    'Tullibardine Vintage 1993',
    'Bladnoch Vintage 1992-2007 (Signatory)',
    "St George's Chapter 6 (unpeated)",
    'Bladnoch 1993-2009 (G&M)',
    'Glenrothes Alba Reserve',
    'Highland Park Harald',
    'Shelter Point Classic Single Cask (KWM) Single Malt Whisky',
    'Auchentoshan Virgin Oak',
    'Tyrconnell Single Malt Irish Whiskey',
    'Glenglassaugh Evolution',
    'Glenrothes Bourbon Cask Reserve',
    'Macallan Gold',
    'Nikka Gold & Gold',
    'Shelter Point Artisanal Single Malt Whisky',
    'Tomatin Legacy',
    'Cardhu Amber Rock',
    'Deanston Virgin Oak',
    'Kilbeggan Irish Reserve Malt Whiskey',
    'Knappogue Castle Vintage',
    'Mackmyra Midnattssol',
    'Tullibardine Aged Oak Edition',
    'Tomatin Cu Bocan',
    'Game of Thrones House Targaryen Cardhu Gold Reserve',
    'Lohin McKinnon Single Malt',
    'Auchentoshan Classic',
    'White Oak Akashi Single Malt (NAS)',
    'Nantou (Omar) Yushan Blended Malt',
    'Glen Moray Classic',
    "Glen Grant The Major's Reserve",
    'Mackmyra Mack',
    'BenRiach Heart of Speyside',
    'Macallan Gold Double Cask',
    'Auchentoshan American Oak',
    'Loch Lomond NAS',
    'Tullibardine Sovereign',
    "McClelland's Highland Single Malt",
    "McClelland's Lowland Single Malt",
    'Port Ellen (all OB releases)',
    'Compass Box Flaming Heart 2018 6th Edition',
    'Compass Box Flaming Heart 2008 2nd Edition',
    'Compass Box Flaming Heart 2015 5th Edition - 15th Anniversary',
    'Highland Park Sigurd',
    'Two Brewers Release 07 Peated',
    'Bruichladdich Port Charlotte 2010 MRC:01',
    'Compass Box Flaming Heart (all editions)',
    'Bruichladdich Port Charlotte PC10 Tro Na Linntean',
    'Glen Garioch 1994',
    'Tomatin Cu Bocan 1989 Limited Edition',
    'Kilchoman Bourbon Single Cask',
    'Glen Garioch 1991',
    'Bruichladdich Port Charlotte 2007 CC:01',
    'Bruichladdich Port Charlotte PC11 Eorna Na H-Alba',
    'Compass Box Flaming Heart 2010 3rd Edition - 10th Anniversary',
    'Talisker 57 North',
    'Compass Box Flaming Heart 2012 4th Edition',
```

```
'Kavalan Distillery Reserve Peaty Cask',
"Talisker Distiller's Edition (all editions)",
'Glen Garioch 1995',
'Compass Box The Lost Blend',
'Bowmore Vault Edition Second Release',
'Bruichladdich Port Charlotte PC12 Oileanach Furachail',
'Wemyss Malts The Rockpool',
'Bruichladdich Port Charlotte PC10 (Second Edition)',
'Spirit of Hven Seven Stars No. 5 Alioth',
'Bowmore Mizunara Cask Finish',
'Dun Bheagan Islay 2009',
'Glenlivet Nadurra Peated Cask Finish',
'Hakushu Single Malt Heavily Peated',
'Bruichladdich Infinity Third Edition',
'Westland Peat Week',
'Game of Thrones House Greyjoy Talisker Select Reserve',
'Bruichladdich Port Charlotte An Turas Mor',
'Nikka Pure Malt Black',
'Two Brewers Peated (all releases)',
'Ben Nevis Celebrated Traditional (NAS)',
'GlenDronach Peated Port Wood',
'Bruichladdich Port Charlotte Scottish Barley Heavily Peated',
'Kilchoman Sanaig',
'Paul John Peated Select Cask',
'Spirit of Hven Tycho’s Star',
'Compass Box Lady Luck',
'Glenglassaugh Torfa',
'Nikka Pure Malt White',
'Paul John Bold',
'Kilchoman Sauternes Cask Matured',
'Big Peat (Douglas Laing)',
'Lohin McKinnon Peated',
'Bruichladdich Islay Barley (all vintages)',
'Mackmyra Svensk Rök',
'Talisker Dark Storm',
'Compass Box Eleuthera',
'Talisker Storm',
'GlenDronach Peated',
'Matsui The Peated',
'Arran Malt Machrie Moor Cask Strength',
'Dun Bheagan Islay (all vintage editions)',
'Game of Thrones The Night’s Watch Oban Bay Reserve',
'Mars Kogamatake The Revival 2011',
'Westland Peated',
'Nikka Yoichi NAS',
'Bowmore Vault Edition First Release',
'Kilchoman Coull Point',
'Elements of Islay "Peat"',
'Paul John Edited',
'BenRiach Peated Quarter Casks',
'Spirit of Hven Seven Stars No. 2 Merak',
'High Coast (Box) Early Days (batches 01/02)',
'Ardmore Traditional Cask',
'Glenfiddich Vintage Cask',
'High Coast (Box) Dalvve',
'Spirit of Hven Seven Stars No. 3 Phecda',
'Two Brewers Release 03 Peated',
'Glen Garioch 1997',
'Dun Bheagan Islay 1999',
'Teerenpeli Suomi 100 Juhlaviski',
'Oban Little Bay',
'Glenfiddich Fire & Cane Experimental Series No. 4',
'Fettercairn Fior',
'Wemyss Malts Peat Chimney',
'Glen Moray Classic Peated',
'Glenglassaugh Octaves Peated',
'Bowmore No.1',
'Benromach Sassicaia',
'Bowmore Gold Reef',
'Penderyn Peated',
'Springbank CV',
'Jura Superstition',
```

```
    'Penderyn Celt',
    'BenRiach Birnie Moss',
    'Spirit of Hven Seven Stars No. 1 Dubhe',
    'Bowmore Small Batch',
    'Glenrothes Peated Cask Reserve',
    'Connemara Peated Single Malt',
    "St George's Chapter 9 (peated)",
    'Highland Park Einar',
    'Old Ballantruan Peated (Toumintoul)',
    'Arran Malt Machrie Moor Peated (all editions)',
    'Bowmore Legend',
    'Bruichladdich Octomore 10 (Third Edition)',
    'Ardbeg Uigeadail',
    'Laphroaig Cairdeas 2015',
    'Bruichladdich Octomore 7.3',
    'Bruichladdich Octomore 6.2',
    'Bruichladdich Octomore 8.3',
    'Amrut Peated Cask Strength',
    'Ardbeg Supernova 2014',
    'Bruichladdich Octomore 9.3',
    'Ardbeg Corryvreckan',
    'Amrut Portpipe Peated Single Cask #4668 (2017)',
    'Ardbeg Alligator',
    'Bruichladdich Octomore 8.2',
    "Lagavulin Distiller's Edition (All Vintages)",
    'Ardbeg Ardbog',
    'Ardbeg Supernova 2015',
    'Compass Box No Name',
    'Kilchoman 2008 Vintage',
    'Bruichladdich Octomore 11.3',
    'Bruichladdich Octomore 7.1',
    'Bruichladdich Octomore 10 (Second Edition)',
    'Compass Box No Name (all editions)',
    'Compass Box Peat Monster 2014 - 10th Anniversary',
    'Laphroaig Cairdeas 2019 Triple Wood Cask Strength',
    'Bruichladdich Octomore 8.4',
    "Ichiro's Malt Chichibu The Peated",
    'Laphroaig Cairdeas 2014 Amontillado',
    'Laphroaig Quarter Cask',
    'Smogen Single Cask (all editions)',
    'Kilchoman 2009 Vintage',
    'Kilchoman Original Cask Strength',
    'Amrut Portpipe Peated Single Cask #2712 (2013)',
    'AnCnoc Cutter',
    'Kilchoman 2007 Vintage',
    'Bruichladdich Octomore 10',
    'Compass Box Peat Monster 2008 Reserve Edition',
    'High Coast (Box) The Festival 2014',
    'Ardbeg Dark Cove',
    'Bruichladdich Octomore 11.1',
    'Amrut 100 Peated',
    'Bruichladdich Octomore 7.2',
    'AnCnoc Rutter',
    'Ardbeg Grooves',
    'Compass Box No Name No. 2',
    'Bruichladdich Octomore 8.1',
    'Ardbeg Supernova 2019',
    'Bruichladdich Octomore 10.1',
    'Laphroaig Cairdeas 2018 Fino',
    'Compass Box Monster 2004',
    'Laphroaig Cairdeas 2016 Madeira',
    'Ardbeg Auriverdes',
    'Bruichladdich Octomore 6.3',
    'Laphroaig An Cuan Mor',
    'Wemyss Malts Kiln Embers',
    'Ardbeg An Oa',
    'Longrow CV',
    'Bruichladdich Octomore 6.1',
    'Bunnahabhain Ceòbanach',
    'Compass Box Peat Monster 2015 Cask Strength',
    'Ardbeg Kelpie',
    'Ardbeg Perpetuum',
```

```
    'Kilchoman Machir Bay (all vintages)',
    'Bruichladdich Port Charlotte Islay Barley Heavily Peated',
    'Amrut Portpipe Peated Single Cask #2713 (2013)',
    'Amrut Portpipe Peated Single Cask (all casks)',
    'Bruichladdich Octomore 9.1',
    'Longrow Peated',
    'Two Brewers Release 12 Peated',
    'Connemara Turf Mor',
    'Kilchoman 100% Islay (all editions)',
    'Compass Box Peat Monster (all editions)',
    'Big Peat Christmas Edition (all editions)',
    'Bruichladdich Octomore 7.4',
    'Bunnahabhain Toiteach A Dha',
    'Compass Box Peat Monster 2005',
    'Compass Box Peat Monster 2006',
    'Compass Box Peat Monster 2012',
    'Kilchoman Spring 2011 Release',
    'Laphroaig Cairdeas 2017',
    'Compass Box Peat Monster 2015',
    "Caol Ila Distiller's Edition (all editions)",
    'Laphroaig Lore',
    'Amrut Peated',
    'Laphroaig Triple Wood',
    'Ardbeg Drum',
    'Ileach Peated Islay Cask Strength',
    'Jura Prophecy',
    'Amrut Portpipe Peated Single Cask #2712 (2016)',
    'Bunnahabhain Toiteach',
    'AnCnoc Flaughter',
    'Benromach Peat Smoke',
    'Bruichladdich Port Charlotte The Peat Project',
    'Kilchoman Winter 2010 Release',
    'Compass Box Peat Monster 2010',
    'Finlaggan Old Reserve',
    'Ileach Peated Islay',
    'Bunnahabhain Cruach Mhona',
    'Bruichladdich Octomore 10.4',
    'Laphroaig QA Cask',
    'Milstone Peated',
    'Tomintoul Peaty Tang',
    'Santis Edition Dreifaltigkeit',
    "McClelland's Islay Single Malt",
    'Santis Edition Dreifaltigkeit / Cask Strength Peated',
    'Santis Cask Strength Peated',
    'Compass Box The General',
    'Compass Box Hedonism Quindecimus',
    'Compass Box The Circus',
    'Chivas Regal Ultis',
    'Compass Box Juveniles',
    'The Irishman Cask Strength',
    'Writers Tears Pot Still Cask Strength',
    'High West Campfire',
    'Compass Box Hedonism The Muse',
    'Hankey Bannister Heritage',
    'Compass Box The Circle',
    'Powers Three Swallow',
    "Compass Box Great King St Artist's Blend",
    'Johnnie Walker Blue Label',
    'Compass Box Great King St Glasgow Blend',
    "Jameson Cooper's Croze Irish Whiskey",
    'Jameson Round Irish Whiskey',
    'Johnnie Walker Explorer’s Club The Gold Route',
    "Ichiro's Malt & Grain World Blended",
    'Powers Signature Release',
    'Cutty Sark Prohibition',
    'Teeling Single Grain (Wine Cask Finish)',
    'Writers Tears Pot Still Irish Whiskey',
    "Compass Box Delilah's",
    'Compass Box Hedonism',
    'Jameson Gold Reserve',
    'Té Bheag',
    "Jameson Blender's Dog Irish Whiskey",
```

```
    'Johnnie Walker Platinum Label',
    'Jameson Select Reserve (Black Barrel)',
    'Nikka Coffey Grain',
    'Blandnoch Pure Scot',
    'Johnnie Walker Explorer’s Club The Spice Road',
    "Compass Box Delilah's XXV",
    'Jameson Signature Reserve',
    'Johnnie Walker Explorer’s Club The Royal Route',
    'Kirin 50% Blend (Fuji Gotemba)',
    'The Irishman Founder’s Reserve',
    "Buchanan's Master Blended",
    'Johnnie Walker Double Black',
    'Compass Box Affinity',
    'Bushmills Red Bush',
    'Suntory Old Whisky',
    'Johnnie Walker Gold Label Reserve',
    'Glendalough Double Barrel',
    'Jameson Crested Irish Whiskey',
    'Jameson Bold Irish Whiskey',
    "Jameson Distiller's Safe Irish Whiskey",
    'Lohin McKinnon Barley and Rye Lightly Peated',
    'Nikka Blended',
    'Compass Box Asyla',
    'Nikka Days',
    'Black Bottle (pre-2013)',
    'Nikka Super Nikka',
    'The Irishman Original Clan Irish Whiskey',
    'Shelter Point Montfort Lot 141 Reserve',
    "Bain's Cape Mountain Whisky",
    'Chivas Regal Mizunara',
    'Powers Gold Label',
    'Teeling Poitin',
    'Famous Jubilee',
    'Suntory The Chita Single Grain',
    "Buchanan's Red Seal Blended",
    'Cutty Sark Storm',
    'Jameson Caskmates Stout Edition',
    "Catto's Rare Old",
    'Black Bottle (after 2013 re-launch)',
    'Kakubin Yellow Label (Suntory Whisky)',
    "Grant's Blended Sherry Cask",
    'Suntory Toki',
    'Famous Grouse Smoky Black (Black Grouse)',
    'West Cork Original Bourbon Cask',
    'Grand Macnish',
    'Hankey Bannister Original',
    "Teacher's Highland Cream",
    'Highland Queen',
    'Jameson Lively Irish Whiskey',
    'Johnnie Walker Explorer’s Club The Adventurer',
    'Jameson Irish Whiskey',
    'Tullamore Dew Original Blended',
    '2 Gingers Irish Whiskey',
    'John Barr Reserve (Black Label)',
    "Grant's Family Reserve Blended",
    'Bushmills Original Blended',
    'The Quiet Man Traditional Irish Whiskey',
    'Whyte & Mackay Special Blended',
    'Cutty Sark',
    "Ballantine's Finest",
    'Famous Grouse',
    "Bell's Original Scotch Whisky",
    'Compass Box Orangerie',
    'Mackmyra Vit Hund',
    'Johnnie Walker White Walker (Game of Thrones)',
    'White Oak Akashi Blended',
    "Dewar's White Label",
    'Johnnie Walker Red Label',
    'P&M Blended Whisky',
    'Passport Blended Scotch',
    'Whyte & Mackay Blended Triple Matured',
    'J&B Rare',
```

```
    'Yamazakura Blended Whisky',
    'Mister Sam Tribute Whisky',
    "Booker's Rye",
    'Lot 40 Cask Strength (Single Cask)',
    'Thomas H. Handy Sazerac',
    'Smooth Ambler Old Scout Single Barrel Rye',
    "J.P. Wiser's Seven Rebels",
    'High West Midwinter Night’s Dram Rye',
    "J.P. Wiser's Dissertation",
    "Michter's Barrel Strength Rye",
    'Little Book Chapter 2 Noe Simple Task',
    "Wild Turkey Master's Keep Cornerstone Rye",
    "J.P. Wiser's Legacy",
    'High West Rendezvous Rye (pre-2018)',
    'Gooderham & Worts Eleven Souls Four Grain (2018)',
    'Shelter Point Single Cask Rye',
    'High West Rendezvous Rye (all bottlings)',
    'Lot 40',
    'Lot 40 Dark Oak',
    "J.P. Wiser's Union 52",
    'Forty Creek Unity',
    'High West Double Rye (new recipe, post-2018)',
    'Whistlepig The Boss Hog',
    'Little Book Chapter 3 The Road Home',
    'Little Book (all Chapters)',
    'Forty Creek Port Wood Reserve 2011/2012',
    'Pikesville Straight Rye',
    'Forty Creek Confederation Oak (Batch A, B)',
    'Lot 40 Cask Strength Third Edition (2019)',
    'Alberta Premium Cask Strength Rye (all batches)',
    'Alberta Premium Cask Strength Rye (Batch 1 2019)',
    'Amrut Rye',
    'Crown Royal Hand Selected Barrel',
    'Forty Creek Confederation Oak (Batch J, K, L)',
    'Forty Creek Heart of Gold',
    'High West Double Rye Manhattan Barrel',
    "J.P. Wiser's Red Letter",
    'Willett Family Estate Rye (all ages)',
    'Wayne Gretzky No. 99 Ninety Nine Proof',
    'High West Bourye',
    'High West Double Rye (all bottlings)',
    'High West Double Rye (pre-2018)',
    'Barrell Rye (all Batches)',
    'Forty Creek Confederation Oak (All Batches)',
    "Jack Daniel's Single Barrel Rye",
    'Angel’s Envy Rye (Rum-finished)',
    'Forty Creek Double Barrel Reserve',
    'FEW Rye Whisky',
    'Alberta Rye Dark Batch',
    'Colonel E.H. Taylor Straight Rye',
    'Forty Creek Confederation Oak (Batch G, H, I)',
    'Crown Royal Noble Collection Wine Barrel Finished',
    'Little Book Chapter 1 The Easy',
    "J.P. Wiser's One Fifty",
    'Crown Royal Monarch 75th Anniversary',
    "J.P. Wiser's Canada 2018",
    'Alberta Premium Dark Horse',
    'High West Yippee Ki-Yay',
    'Rittenhouse Rye 100 Proof',
    'Forty Creek Victory',
    'Gooderham & Worts Four Grain',
    "Michter's Single Barrel Straight Rye",
    "Crown Royal Blender's Select",
    'Wild Turkey 101 Rye',
    'Forty Creek Evolution',
    'Forty Creek Copper Pot Reserve',
    'Sazerac Straight Rye',
    "J.P. Wiser's Wheatfield Gold",
    'Crown Royal Northern Harvest Rye',
    'Stalk & Barrel Rye',
    'Crown Royal XO',
    'High West Double Rye Campfire Barrel',
```

```
"J.P. Wiser's Small Batch",
'Crown Royal Noble Collection French Oak Cask Finished',
"J.P. Wiser's Triple Barrel Rye",
'Barrell Rye Batch 002',
'Forty Creek Heritage 2017',
'Knob Creek Small Batch Straight Rye Whiskey',
'Caribou Crossing Single Barrel',
'Crown Royal Noble Collection Cornerstone Blend',
'Forty Creek Confederation Oak (Batch E, F)',
'Millstone 100 Rye',
'George Dickel Rye',
'Shelter Point Artisanal Cask Strength Whisky',
'Prichard’s Rye',
'High West Son of Bourye',
'Forty Creek Confederation Oak (Batch C, D)',
'Crown Royal Reserve',
'Jim Beam Pre-Prohibition Rye',
'Stalk & Barrel Rye (Cask Strength)',
"Basil Hayden's Rye Whiskey",
'Bulleit Rye',
"Potter's Special Old",
"Forty Creek Founder's Reserve",
'Woodford Reserve Straight Rye',
'Crown Royal Limited Edition',
"J.P. Wiser's Double Still Rye",
'Willett Family Estate Rye XCF 1.0',
'Old Overholt Bonded',
'Canadian Club 100% Rye',
'Forty Creek Barrel Select',
'Wayne Gretzky No. 99 Ice Cask',
'Yellow Rose Straight Rye',
"Jack Daniel's Rested Tennessee Rye (Batch 1/2)",
'Crown Royal Black',
'Forty Creek Three Grain Harmony',
'Stalk & Barrel Red Blend',
'66 Gilead Crimson Rye',
'Stalk & Barrel 11+1 Canadian whisky',
"Crown Royal Bourbon Mash (Blender's Mash)",
'Canadian Club Sherry Cask',
'Pendleton 1910',
'Wild Turkey 81 Rye',
'Hiram Walker Special Old Rye',
'Koval Single Barrel Rye',
'Century Reserve Lot 15/25',
'Alberta Premium',
"J.P. Wiser's Hopped",
'Canada Gold',
'Royal Canadian Small Batch',
'Ezra Brooks Rye',
"J.P. Wiser's Deluxe",
'Basil Hayden’s Dark Rye',
'Basil Hayden’s Two by Two Rye',
'Canadian Mist Black Diamond',
'Coyote Ugly',
'Jim Beam Rye',
'Collingwood',
'Stalk & Barrel Blue Blend',
"Gibson's Finest Sterling",
'Templeton Rye',
"J.P. Wiser's Rye",
'Wayne Gretzky No. 99 Red Cask',
"Pendleton (Let'er Buck)",
'Canadian Club Barley Batch',
'Twelve Barrels',
'Basil Hayden’s Caribbean Reserve Rye',
'Pendleton Midnight',
'Old Overholt',
'Schenley Golden Wedding',
'Rich and Rare Reserve',
'Rich and Rare',
'Silk Tassel',
'Crown Royal',
```

```
 "Seagram's VO",
 'Canadian Mist',
 '8 Seconds',
 "J.P. Wiser's Special Blend",
 'Canadian Club (Premium)',
 "Seagram's Canadian 83",
 'Proof Whisky',
 'Black Velvet Deluxe',
 'Barrell Single Barrel (all barrels)',
 'George T Stagg',
 'William Larue Weller',
 'Parker’s Heritage 6th Blend of Mashbills',
 'Stagg Jr batch 9 (131.9 proof)',
 'Four Roses Small Batch Limited Edition',
 'Barrell Bourbon Batch 011',
 'Stagg Jr (batches 3+)',
 'Stagg Jr batch 9 (116.8 proof)',
 'Willett Family Estate Bourbon (all ages)',
 'Barrell Bourbon Batch 019',
 'Parker’s Heritage 1st',
 'Barrell Bourbon Batch 015',
 'Stagg Jr batch 5 (129.7 proof)',
 'William Heavenhill BiB',
 'Stagg Jr batch 4 (132.2 proof)',
 "Wild Turkey Master's Keep Revival",
 'Barrell Bourbon New Years (all Batches)',
 "Blanton's Straight from the Barrel Bourbon",
 'Barrell Bourbon New Years Batch 2018',
 'Barrell Bourbon Batch 018',
 'Colonel EH. Taylor Barrel Proof',
 'Stagg Jr batch 11 (127.9 proof)',
 'Heaven Hill Select Stock Barrel',
 "Jack Daniel's 150th Anniversary",
 'Barrell Bourbon Batch 013',
 'Barrell Dovetail Whiskey (all Batches)',
 "Angel's Envy Cask Strength",
 'Barrell Bourbon New Years Batch 2017',
 "Wild Turkey Master's Keep Decades",
 'Barrell Bourbon Batch 009',
 'Colonel E.H. Taylor Four Grain',
 'Elijah Craig Barrel Proof',
 'Smooth Ambler Old Scout Single Barrel Bourbon',
 'Yellowstone 2018 Limited Edition',
 'Wild Turkey Kentucky Spirit Single Barrel',
 'Colonel E.H. Taylor Single Barrel',
 'Stagg Jr batch 3 (132.1 proof)',
 "Russell's Reserve Single Barrel",
 'Barrell Bourbon (all Batches)',
 'Barrell Bourbon Batch 016',
 'Old Forester 1920 Prohibition Style',
 "J.P. Wiser's Last Barrels",
 'Stagg Jr batch 10 (126.4 proof)',
 "Booker's Small Batch Straight Bourbon",
 'Knob Creek Single Barrel Reserve Bourbon',
 'Wild Turkey Diamond Anniversary',
 'Barrell Bourbon Batch 017',
 'Elmer T. Lee Single Barrel Bourbon',
 'Barrell Bourbon Batch 007',
 ...]
```

So going over the list, I was able to find a few whiskies that contain an age, and didn't follow the convention. First is Octomore, which is pretty pricey brand.

In [51]:

```
df2.loc[df2['Whisky'].str.contains('Octomore')]
```

Out[51]:

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages
1052	Bruichladdich Octomore 10 (Third Edition)	9.25	0.17	3	SingleMalt-like	J	Scotland	Malt	Unknown
1059	Bruichladdich Octomore 7.3	9.10	0.47	14	SingleMalt-like	J	Scotland	Malt	Unknown
1060	Bruichladdich Octomore 6.2	9.08	0.18	12	SingleMalt-like	J	Scotland	Malt	Unknown
1061	Bruichladdich Octomore 8.3	9.08	0.19	13	SingleMalt-like	J	Scotland	Malt	Unknown
1065	Bruichladdich Octomore 9.3	9.07	0.26	12	SingleMalt-like	J	Scotland	Malt	Unknown
1070	Bruichladdich Octomore 8.2	9.03	0.22	8	SingleMalt-like	J	Scotland	Malt	Unknown
1077	Bruichladdich Octomore 11.3	9.01	0.27	4	SingleMalt-like	J	Scotland	Malt	Unknown
1078	Bruichladdich Octomore 7.1	9.01	0.28	17	SingleMalt-like	J	Scotland	Malt	Unknown
1081	Bruichladdich Octomore 10 (Second Edition)	9.00	0.26	12	SingleMalt-like	J	Scotland	Malt	Unknown
1085	Bruichladdich Octomore 8.4	8.98	0.09	4	SingleMalt-like	J	Scotland	Malt	Unknown
1098	Bruichladdich Octomore 10	8.93	0.35	6	SingleMalt-like	J	Scotland	Malt	Unknown
1102	Bruichladdich Octomore 11.1	8.90	0.20	5	SingleMalt-like	J	Scotland	Malt	Unknown
1104	Bruichladdich Octomore 7.2	8.89	0.44	14	SingleMalt-like	J	Scotland	Malt	Unknown
1110	Bruichladdich Octomore 8.1	8.86	0.14	14	SingleMalt-like	J	Scotland	Malt	Unknown
1112	Bruichladdich Octomore 10.1	8.85	0.19	5	SingleMalt-like	J	Scotland	Malt	Unknown
1118	Bruichladdich Octomore 6.3	8.82	0.62	12	SingleMalt-like	J	Scotland	Malt	Unknown
1124	Bruichladdich Octomore 6.1	8.79	0.30	21	SingleMalt-like	J	Scotland	Malt	Unknown
1136	Bruichladdich Octomore 9.1	8.75	0.35	10	SingleMalt-like	J	Scotland	Malt	Unknown
1144	Bruichladdich Octomore 7.4	8.72	0.60	13	SingleMalt-like	J	Scotland	Malt	Unknown
1171	Bruichladdich Octomore 10.4	8.36	0.84	3	SingleMalt-like	J	Scotland	Malt	Unknown

Doing some more research all the Octomores are 5, aside from the Octomore 10s. So let's use our .at and/or another loop to get those right.

In [52]:

```
df2.at[1052, 'Ages']='10'
df2.at[1081, 'Ages']='10'
```

In [53]:

```
df2.loc[df2['Whisky'].str.contains('Octomore')]
```

Out[53]:

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages
1052	Bruichladdich Octomore 10 (Third Edition)	9.25	0.17	3	SingleMalt-like	J	Scotland	Malt	10
1059	Bruichladdich Octomore 7.3	9.10	0.47	14	SingleMalt-like	J	Scotland	Malt	Unknown
1060	Bruichladdich Octomore 6.2	9.08	0.18	12	SingleMalt-like	J	Scotland	Malt	Unknown
1061	Bruichladdich Octomore 8.3	9.08	0.19	13	SingleMalt-like	J	Scotland	Malt	Unknown
1065	Bruichladdich Octomore 9.3	9.07	0.26	12	SingleMalt-like	J	Scotland	Malt	Unknown
1070	Bruichladdich Octomore 8.2	9.03	0.22	8	SingleMalt-like	J	Scotland	Malt	Unknown
1077	Bruichladdich Octomore 11.3	9.01	0.27	4	SingleMalt-like	J	Scotland	Malt	Unknown
1078	Bruichladdich Octomore 7.1	9.01	0.28	17	SingleMalt-like	J	Scotland	Malt	Unknown
1081	Bruichladdich Octomore 10 (Second Edition)	9.00	0.26	12	SingleMalt-like	J	Scotland	Malt	10
1085	Bruichladdich Octomore 8.4	8.98	0.09	4	SingleMalt-like	J	Scotland	Malt	Unknown
1098	Bruichladdich Octomore 10	8.93	0.35	6	SingleMalt-like	J	Scotland	Malt	Unknown
1102	Bruichladdich Octomore 11.1	8.90	0.20	5	SingleMalt-like	J	Scotland	Malt	Unknown
1104	Bruichladdich Octomore 7.2	8.89	0.44	14	SingleMalt-like	J	Scotland	Malt	Unknown
1110	Bruichladdich Octomore 8.1	8.86	0.14	14	SingleMalt-like	J	Scotland	Malt	Unknown

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages
1112	Bruichladdich Octomore 10.1	8.85	0.19	5	SingleMalt-like	J	Scotland	Malt	Unknown
1118	Bruichladdich Octomore 6.3	8.82	0.62	12	SingleMalt-like	J	Scotland	Malt	Unknown
1124	Bruichladdich Octomore 6.1	8.79	0.30	21	SingleMalt-like	J	Scotland	Malt	Unknown
1136	Bruichladdich Octomore 9.1	8.75	0.35	10	SingleMalt-like	J	Scotland	Malt	Unknown
1144	Bruichladdich Octomore 7.4	8.72	0.60	13	SingleMalt-like	J	Scotland	Malt	Unknown
1171	Bruichladdich Octomore 10.4	8.36	0.84	3	SingleMalt-like	J	Scotland	Malt	Unknown

In [54]:

```
for i in df2.index:
    split_name=df2['Whisky'][i].split()
    for item in split_name:
        if item=='Octomore' and df2['Ages'][i]=='Unknown':
            df2.at[i,'Ages']='5'
```

In [55]:

```
df2.loc[df2['Whisky'].str.contains('Octomore')]
```

Out[55]:

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages
1052	Bruichladdich Octomore 10 (Third Edition)	9.25	0.17	3	SingleMalt-like	J	Scotland	Malt	10
1059	Bruichladdich Octomore 7.3	9.10	0.47	14	SingleMalt-like	J	Scotland	Malt	5
1060	Bruichladdich Octomore 6.2	9.08	0.18	12	SingleMalt-like	J	Scotland	Malt	5
1061	Bruichladdich Octomore 8.3	9.08	0.19	13	SingleMalt-like	J	Scotland	Malt	5
1065	Bruichladdich Octomore 9.3	9.07	0.26	12	SingleMalt-like	J	Scotland	Malt	5
1070	Bruichladdich Octomore 8.2	9.03	0.22	8	SingleMalt-like	J	Scotland	Malt	5
1077	Bruichladdich Octomore 11.3	9.01	0.27	4	SingleMalt-like	J	Scotland	Malt	5
1078	Bruichladdich Octomore 7.1	9.01	0.28	17	SingleMalt-like	J	Scotland	Malt	5
1081	Bruichladdich Octomore 10 (Second Edition)	9.00	0.26	12	SingleMalt-like	J	Scotland	Malt	10
1085	Bruichladdich Octomore 8.4	8.98	0.09	4	SingleMalt-like	J	Scotland	Malt	5
1098	Bruichladdich Octomore 10	8.93	0.35	6	SingleMalt-like	J	Scotland	Malt	5
1102	Bruichladdich Octomore 11.1	8.90	0.20	5	SingleMalt-like	J	Scotland	Malt	5
1104	Bruichladdich Octomore 7.2	8.89	0.44	14	SingleMalt-like	J	Scotland	Malt	5
1110	Bruichladdich Octomore 8.1	8.86	0.14	14	SingleMalt-like	J	Scotland	Malt	5
1112	Bruichladdich Octomore 10.1	8.85	0.19	5	SingleMalt-like	J	Scotland	Malt	5
1118	Bruichladdich Octomore 6.3	8.82	0.62	12	SingleMalt-like	J	Scotland	Malt	5
1124	Bruichladdich Octomore 6.1	8.79	0.30	21	SingleMalt-like	J	Scotland	Malt	5
1136	Bruichladdich Octomore 9.1	8.75	0.35	10	SingleMalt-like	J	Scotland	Malt	5
1144	Bruichladdich Octomore 7.4	8.72	0.60	13	SingleMalt-like	J	Scotland	Malt	5
1171	Bruichladdich Octomore 10.4	8.36	0.84	3	SingleMalt-like	J	Scotland	Malt	5

Bruichladdich (who also makes Octomore) didn't use the convention when naming its Port Charlotte brand. They just like to do things differently.

In [56]:

```
df2.loc[df2['Whisky'].str.contains('Charlotte')]
```

Out[56]:

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages
141	Bruichladdich Port Charlotte 2009 MC:01	8.79	0.29	10	SingleMalt-like	C	Scotland	Malt	Unknown
898	Bruichladdich Port Charlotte 2010 MRC:01	8.98	0.20	12	SingleMalt-like	I	Scotland	Malt	Unknown
901	Bruichladdich Port Charlotte PC10 Tro Na Linntean	8.96	0.39	12	SingleMalt-like	I	Scotland	Malt	Unknown
910	Bruichladdich Port Charlotte 2007 CC:01	8.93	0.29	17	SingleMalt-like	I	Scotland	Malt	Unknown
911	Bruichladdich Port Charlotte PC11 Eorna Na H-Alba	8.93	0.26	6	SingleMalt-like	I	Scotland	Malt	Unknown
925	Bruichladdich Port Charlotte PC12 Oileanach Fu...	8.87	0.43	13	SingleMalt-like	I	Scotland	Malt	Unknown
928	Bruichladdich Port Charlotte 10yo Heavily Peat...	8.85	0.36	12	SingleMalt-like	I	Scotland	Malt	10
930	Bruichladdich Port Charlotte PC10 (Second Edit...	8.84	0.19	7	SingleMalt-like	I	Scotland	Malt	Unknown
942	Bruichladdich Port Charlotte 10yo Heavily Peat...	8.77	0.21	9	SingleMalt-like	I	Scotland	Malt	10
950	Bruichladdich Port Charlotte An Turas Mor	8.74	0.27	13	SingleMalt-like	I	Scotland	Malt	Unknown
959	Bruichladdich Port Charlotte Scottish Barley H...	8.70	0.26	23	SingleMalt-like	I	Scotland	Malt	Unknown
1133	Bruichladdich Port Charlotte Islay Barley Heav...	8.76	0.18	8	SingleMalt-like	J	Scotland	Malt	Unknown
1165	Bruichladdich Port Charlotte The Peat Project	8.51	0.38	7	SingleMalt-like	J	Scotland	Malt	Unknown

In [57]:

```
df2.at[901, 'Ages']='10'
df2.at[911, 'Ages']='11'
df2.at[925, 'Ages']='12'
df2.at[930, 'Ages']='10'
```

Let's see if we can resolve the rest of the unknowns based on the type.

In [58]:

```
df1['Type'].value_counts()
```

Out[58]:

```
Malt         1158
Blend         302
Bourbon       209
Rye            84
Grain           7
Wheat           2
Flavoured       1
Barley          1
Whiskey         1
Name: Type, dtype: int64
```

Upon a little research, I've found that the minimum age for whiskies is typically 2-3 years. So I'll make a list of whiskies that must be at least 3 years old. Then make a loop that'll go through the dataframe: if the age is 'Unknown' and it corresponds to a member of this list, it'll make it a '3', otherwise, it'll make it a '2'.

In [59]:

```
yr3=['Malt','Blend','Grain','Barley']
```

In [60]:

```
for a in df2['Whisky'].index:
    if df2['Ages'][a] == 'Unknown' and df1['Type'][a] in yr3:
        df2.at[a,'Ages']='3'
    elif df2['Ages'][a] == 'Unknown' and df1['Type'][a] not in yr3:
        df2.at[a,'Ages']='2'
```

In [61]:

```
df2.head()
```

Out[61]:

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages
0	Macallan 10yo Full Proof 57% 1980 (OB, Giovine...	9.57	0.24	3	SingleMalt-like	A	Scotland	Malt	10
1	Ledaig 42yo Dusgadh	9.48	0.23	3	SingleMalt-like	C	Scotland	Malt	42
2	Laphroaig 27yo 57.4% 1980-2007 (OB, 5 Oloroso ...	9.42	0.23	4	SingleMalt-like	C	Scotland	Malt	27
3	Glenfarclas 40yo	9.29	0.26	17	SingleMalt-like	A	Scotland	Malt	40
4	Glengoyne 25yo	9.24	0.22	21	SingleMalt-like	A	Scotland	Malt	25

In [62]:

```
df2['Ages'].unique()
```

Out[62]:

```
array(['10', '42', '27', '40', '25', '3', '30', '12', '18', '21', '8',
       '16', '17', '20', '19', '15', '35', '13', '14', '11', '22', '26',
       '23', '5', '7', '24', '31', '32', '28', '37', '29', '9', '4', '2',
       '41', '6'], dtype=object)
```

Now we have this Ages no longer has any unknowns. Now we can deal with the rest of the nulls in the data.

In [63]:

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1765 entries, 0 to 1766
Data columns (total 9 columns):
Whisky         1765 non-null object
Meta Critic    1765 non-null object
STDEV          1765 non-null object
#              1765 non-null int64
Class          1765 non-null object
Cluster        1455 non-null object
Country        1765 non-null object
Type           1765 non-null object
Ages           1765 non-null object
dtypes: int64(1), object(8)
memory usage: 217.9+ KB
```

So barring any other invalid values, it looks like only 'Cluster' has nulls, 310 of them to be exact.

In [64]:

```
df2['Cluster'].unique()
```

Out[64]:

```
array(['A', 'C', 'B', 'F', 'E', 'G', 'H', 'I', 'J', nan, 'R4', 'R2', 'R1',
       'R0', 'R3'], dtype=object)
```

In [65]:

```
df2.loc[df2['Cluster'].isnull()]
```

Out[65]:

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages
1180	Compass Box The General	9.21	0.28	11	Scotch-like	NaN	Scotland	Blend	3
1181	Black Bull 40yo	9.09	0.35	11	Scotch-like	NaN	Scotland	Blend	40
1182	Compass Box Hedonism Quindecimus	8.92	0.39	8	Scotch-like	NaN	Scotland	Blend	3
1183	Compass Box The Circus	8.84	0.29	8	Scotch-like	NaN	Scotland	Blend	3
1184	Powers 12yo John's Lane	8.84	0.36	19	Scotch-like	NaN	Ireland	Blend	12
...	...	...	...	...	...	...	...	...	...
1675	Barrell Whiskey (all Batches)	8.51	0.47	7	Bourbon-like	NaN	USA	Blend	3
1684	Barrell Infinity Barrel Project (all releases)	8.49	0.34	5	Bourbon-like	NaN	USA	Blend	3
1686	Barrell Whiskey Batch 005	8.48	0.62	3	Bourbon-like	NaN	USA	Blend	3
1753	66 Gilead The Wild Oak	7.91	0.57	7	Bourbon-like	NaN	Canada	Blend	3
1764	Jim Beam Red Stag (Black Cherry)	7.35	1.01	4	Bourbon-like	NaN	USA	Flavoured	2

310 rows × 9 columns

My gut tells me the easiest way to fix this is to just make a new cluster category, 'U' for undefined (or unknown).

In [66]:

```
df2["Cluster"].fillna("U", inplace = True)
```

In [67]:

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1765 entries, 0 to 1766
Data columns (total 9 columns):
Whisky         1765 non-null object
Meta Critic    1765 non-null object
STDEV          1765 non-null object
#              1765 non-null int64
Class          1765 non-null object
Cluster        1765 non-null object
Country        1765 non-null object
Type           1765 non-null object
Ages           1765 non-null object
dtypes: int64(1), object(8)
memory usage: 217.9+ KB
```

Since 5 of these remaining columns are categorical variables, I think we need to do some getdummies. Since it works better with string variables.

In [68]:

```
df2.head()
```

Out[68]:

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type	Ages
0	Macallan 10yo Full Proof 57% 1980 (OB, Giovine...	9.57	0.24	3	SingleMalt-like	A	Scotland	Malt	10
1	Ledaig 42yo Dusgadh	9.48	0.23	3	SingleMalt-like	C	Scotland	Malt	42
2	Laphroaig 27yo 57.4% 1980-2007 (OB, 5 Oloroso ...	9.42	0.23	4	SingleMalt-like	C	Scotland	Malt	27
3	Glenfarclas 40yo	9.29	0.26	17	SingleMalt-like	A	Scotland	Malt	40
4	Glengoyne 25yo	9.24	0.22	21	SingleMalt-like	A	Scotland	Malt	25

I'm pretty sure I don't want that name in when I do the modeling, so let's drop it.

In [69]:

```
df2.drop('Whisky',axis=1,inplace=True)
```

In [70]:

```
cat=['Class','Cluster','Country','Type']
```

In [71]:

```
dums = pd.get_dummies(df2[cat], drop_first=False)
```

In [72]:

```
dums.head()
```

Out[72]:

	Class_Bourbon-like	Class_Rye-like	Class_Scotch-like	Class_SingleMalt-like	Cluster_A	Cluster_B	Cluster_C	Cluster_E	Cluster_F	Clust
0	0	0	0	1	1	0	0	0	0	
1	0	0	0	1	0	0	1	0	0	
2	0	0	0	1	0	0	1	0	0	
3	0	0	0	1	1	0	0	0	0	
4	0	0	0	1	1	0	0	0	0	

5 rows × 45 columns

In [73]:

```
df2.drop(df2[cat],axis=1,inplace=True)
```

In [74]:

```
X=df2.merge(dums,right_index=True,left_index=True)
```

In [75]:

```
X.describe()
```

Out[75]:

	#	Class_Bourbon-like	Class_Rye-like	Class_Scotch-like	Class_SingleMalt-like	Cluster_A	Cluster_B	Cluster_C	
count	1765.000000	1765.000000	1765.000000	1765.000000	1765.000000	1765.000000	1765.000000	1765.000000	1
mean	11.092351	0.127479	0.128612	0.076487	0.667422	0.058924	0.029462	0.124646	
std	6.967285	0.333603	0.334865	0.265851	0.471270	0.235548	0.169145	0.330411	
min	3.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	5.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	9.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	
75%	15.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	
max	34.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

8 rows × 46 columns

In [76]:

```
X.head()
```

Out[76]:

	Meta Critic	STDEV	#	Ages	Class_Bourbon-like	Class_Rye-like	Class_Scotch-like	Class_SingleMalt-like	Cluster_A	Cluster_B	...	Country_
0	9.57	0.24	3	10	0	0	0	1	1	0	...	
1	9.48	0.23	3	42	0	0	0	1	0	0	...	
2	9.42	0.23	4	27	0	0	0	1	0	0	...	
3	9.29	0.26	17	40	0	0	0	1	1	0	...	
4	9.24	0.22	21	25	0	0	0	1	1	0	...	

5 rows × 49 columns

Now I need to make sure all my columns contain numbers, or else I'll get some errors when I start modeling.

In [77]:

```
X['Meta Critic']= X['Meta Critic'].astype(float)
```

In [78]:

```
X['STDEV']=X['STDEV'].astype(float)
```

In [79]:

```
X['Ages']=X['Ages'].astype(int)
```

In [80]:

```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1765 entries, 0 to 1766
Data columns (total 49 columns):
Meta Critic                1765 non-null float64
STDEV                      1765 non-null float64
#                          1765 non-null int64
Ages                       1765 non-null int32
Class_Bourbon-like         1765 non-null uint8
Class_Rye-like             1765 non-null uint8
Class_Scotch-like          1765 non-null uint8
Class_SingleMalt-like      1765 non-null uint8
Cluster_A                  1765 non-null uint8
Cluster_B                  1765 non-null uint8
Cluster_C                  1765 non-null uint8
Cluster_E                  1765 non-null uint8
Cluster_F                  1765 non-null uint8
Cluster_G                  1765 non-null uint8
Cluster_H                  1765 non-null uint8
Cluster_I                  1765 non-null uint8
Cluster_J                  1765 non-null uint8
Cluster_R0                 1765 non-null uint8
Cluster_R1                 1765 non-null uint8
Cluster_R2                 1765 non-null uint8
Cluster_R3                 1765 non-null uint8
Cluster_R4                 1765 non-null uint8
Cluster_U                  1765 non-null uint8
Country_Belgium            1765 non-null uint8
Country_Canada             1765 non-null uint8
Country_England            1765 non-null uint8
Country_Finland            1765 non-null uint8
Country_France             1765 non-null uint8
Country_India              1765 non-null uint8
Country_Ireland            1765 non-null uint8
Country_Japan              1765 non-null uint8
Country_Netherlands        1765 non-null uint8
Country_Scotland           1765 non-null uint8
```

```
Country_South Africa     1765 non-null uint8
Country_Sweden           1765 non-null uint8
Country_Switzerland      1765 non-null uint8
Country_Taiwan           1765 non-null uint8
Country_Tasmania         1765 non-null uint8
Country_USA              1765 non-null uint8
Country_Wales            1765 non-null uint8
Type_Barley              1765 non-null uint8
Type_Blend               1765 non-null uint8
Type_Bourbon             1765 non-null uint8
Type_Flavoured           1765 non-null uint8
Type_Grain               1765 non-null uint8
Type_Malt                1765 non-null uint8
Type_Rye                 1765 non-null uint8
Type_Wheat               1765 non-null uint8
Type_Whiskey             1765 non-null uint8
dtypes: float64(2), int32(1), int64(1), uint8(45)
memory usage: 219.6 KB
```

I think we've done a nice amount of data processing. I think we're ready to move on to the modeling.

Step 5: Train test split

In [81]:

```
len(y)
```

Out[81]:

```
1765
```

In [82]:

```
y.to_numpy()
```

Out[82]:

```
array([1, 1, 1, ..., 0, 0, 0], dtype=int64)
```

In [83]:

```
y.shape
```

Out[83]:

```
(1765,)
```

In [84]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size= 0.25, random_state=42)
```

Step 6: Start classifying

We'll start with logistic regression

In [85]:

```
logreg = LogisticRegression(fit_intercept=False,C=1e12, solver='liblinear')
```

In [86]:

```
model_log = logreg.fit(X_train, y_train).decision_function(X_test)
```

In [87]:

```
y_hat_train = logreg.predict(X_train)
y_hat_test = logreg.predict(X_test)
```

In [88]:

```
cnf_matrix = confusion_matrix(y_test, y_hat_test)
print('Confusion Matrix:\n', cnf_matrix)
```

```
Confusion Matrix:
 [[277  30]
 [ 65  70]]
```

In [89]:

```
residuals = np.abs(y_train - y_hat_train)
print(pd.Series(residuals).value_counts())
print('------------------------------------')
print(pd.Series(residuals).value_counts(normalize=True))
```

```
0    1118
1     205
dtype: int64
------------------------------------
0    0.845049
1    0.154951
dtype: float64
```

In [90]:

```
residuals = np.abs(y_test - y_hat_test)
print(pd.Series(residuals).value_counts())
print('------------------------------------')
print(pd.Series(residuals).value_counts(normalize=True))
```

```
0    347
1     95
dtype: int64
------------------------------------
0    0.785068
1    0.214932
dtype: float64
```

In [91]:

```
print('Training Precision: ', precision_score(y_train, y_hat_train))
print('Testing Precision: ', precision_score(y_test, y_hat_test))
print('\n')

print('Training Recall: ', recall_score(y_train, y_hat_train))
print('Testing Recall: ', recall_score(y_test, y_hat_test))
print('\n')

print('Training Accuracy: ', accuracy_score(y_train, y_hat_train))
print('Testing Accuracy: ', accuracy_score(y_test, y_hat_test))
print('\n')

print('Training F1-Score: ', f1_score(y_train, y_hat_train))
print('Testing F1-Score: ', f1_score(y_test, y_hat_test))
```

```
Training Precision:  0.7784090909090909
Testing Precision:  0.7


Training Recall:  0.683291770573566
Testing Recall:  0.5185185185185185


Training Accuracy:  0.8450491307634165
Testing Accuracy:  0.7850678733031674


Training F1-Score:  0.7277556440903054
Testing F1-Score:  0.5957446808510639
```

So for a baseline, this isn't terrible. A little underfit for my tastes (that f1 score on the testing data looks bad). So

So for a baseline, this isn't terrible. A little underfit for my tastes (that F1 score on the testing data looks bad). So we'll see how some other models work.

Decision Trees (model #2)

In [92]:

```
tree = DecisionTreeClassifier(criterion='gini')
```

In [93]:

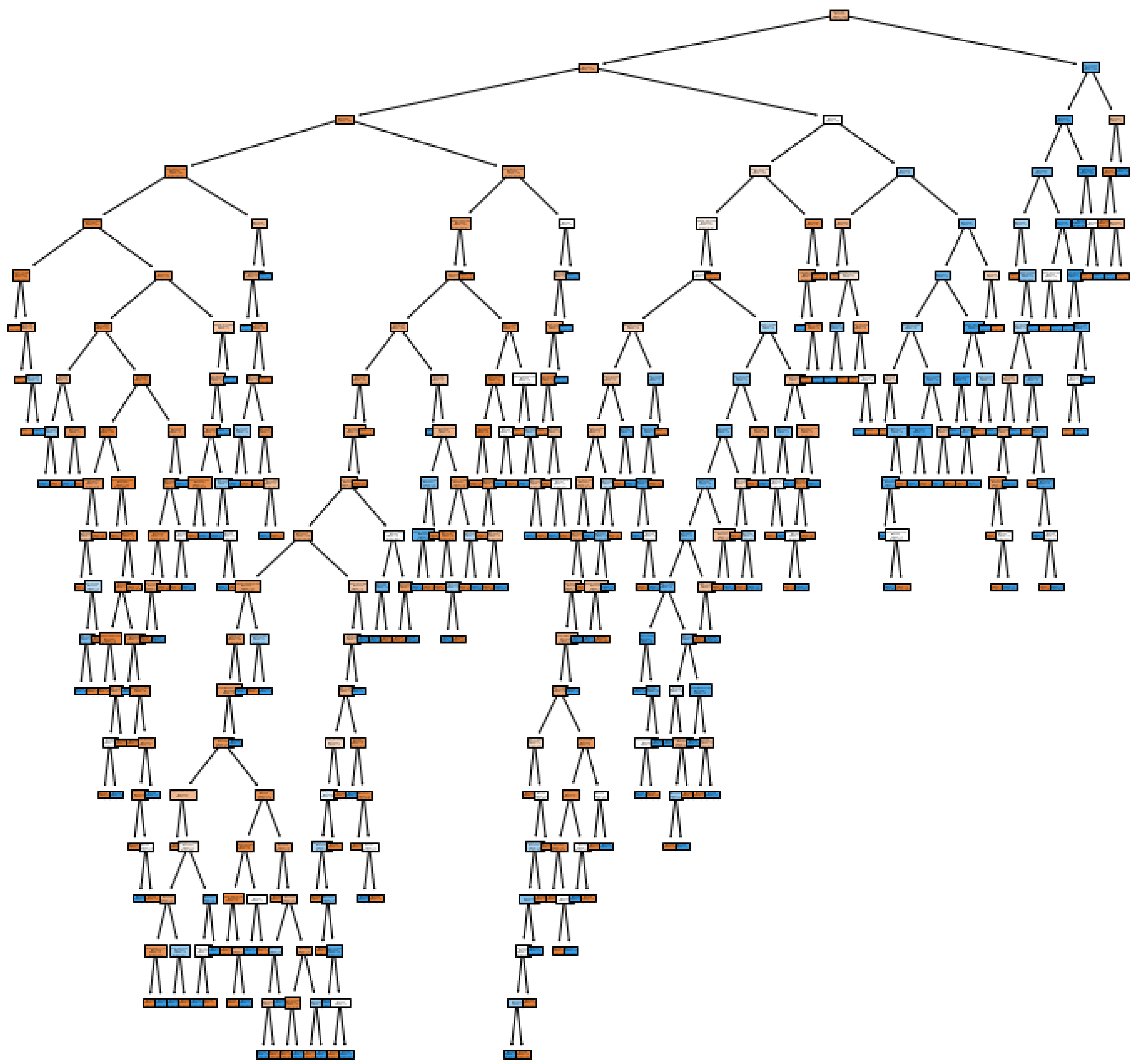
```
tree.fit(X_train, y_train)
```

Out[93]:

```
DecisionTreeClassifier()
```

In [94]:

```
#This cell takes a few minutes to run
plt.figure(figsize=(12,12))
plot_tree(tree, feature_names=X.columns, filled=True)
plt.show()
```



In [95]:

```
tree.score(X_test,y_test)
```

Out[95]:

```
0.7669683257918553
```

In [96]:

```
tree2 = DecisionTreeClassifier(criterion='entropy')
```

In [97]:

```
tree2.fit(X_train, y_train)
```

Out[97]:

```
DecisionTreeClassifier(criterion='entropy')
```

In [98]:

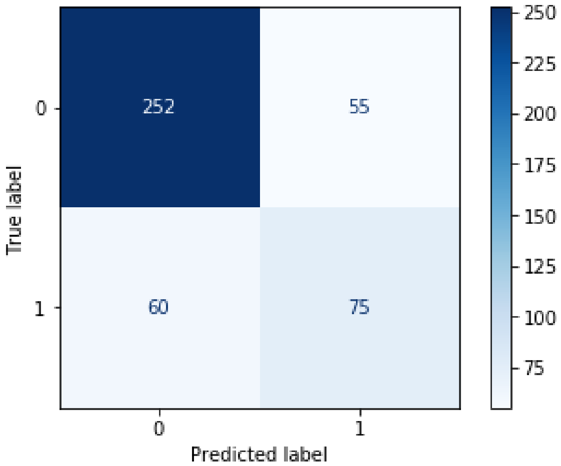
```
tree2.score(X_test,y_test)
```

Out[98]:

```
0.7398190045248869
```

In [99]:

```
plot_confusion_matrix(tree2, X_test, y_test ,cmap=plt.cm.Blues)
plt.show()
```

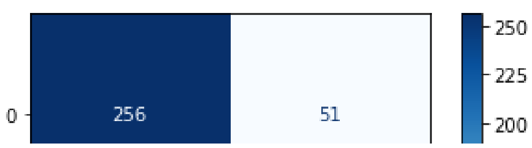


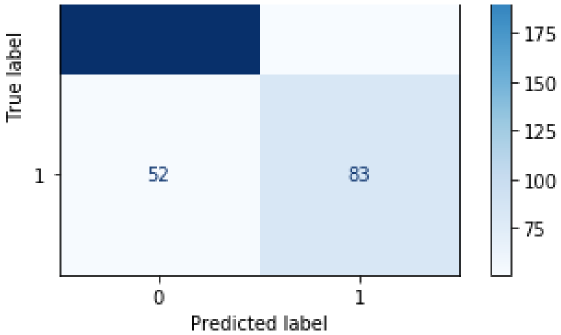
In [100]:

```
#Special thanks to Lindsey Berlin for this code, I added the confusion matrix
def evaluate_model(model, X_train, X_test, y_train, y_test):
    train_preds = model.predict(X_train)
    test_preds = model.predict(X_test)
    plot_confusion_matrix(model, X_test, y_test ,cmap=plt.cm.Blues)
    plt.show()

    train_preds_proba = model.predict_proba(X_train)[:, 1]
    test_preds_proba = model.predict_proba(X_test)[:, 1]

    print('Train Scores:')
    print(f"Accuracy: {model.score(X_train, y_train):.3f}")
    print(f"F1 Score: {f1_score(y_train, train_preds):.3f}")
    print(f"ROC-AUC: {roc_auc_score(y_train, train_preds_proba):.3f}")
    print('Test Scores:')
    print(f"Accuracy: {model.score(X_test, y_test):.3f}")
    print(f"F1 Score: {f1_score(y_test, test_preds):.3f}")
    print(f"ROC-AUC: {roc_auc_score(y_test, test_preds_proba):.3f}")\
```

In [101]:

```
evaluate_model(tree, X_train, X_test, y_train, y_test)
```

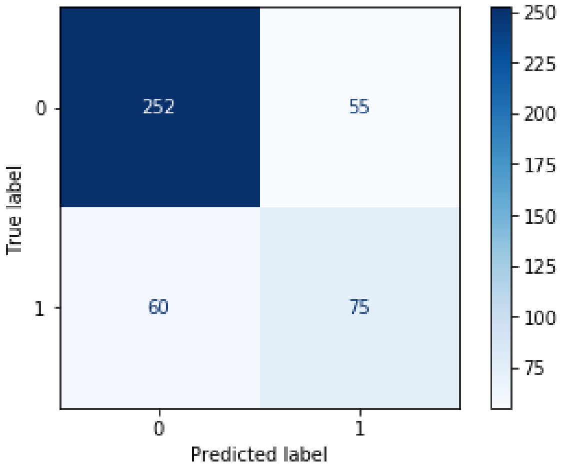





```
Train Scores:
Accuracy: 1.000
F1 Score: 1.000
ROC-AUC: 1.000
Test Scores:
Accuracy: 0.767
F1 Score: 0.617
ROC-AUC: 0.724
```

In [102]:

```
evaluate_model(tree2, X_train, X_test, y_train, y_test)
```

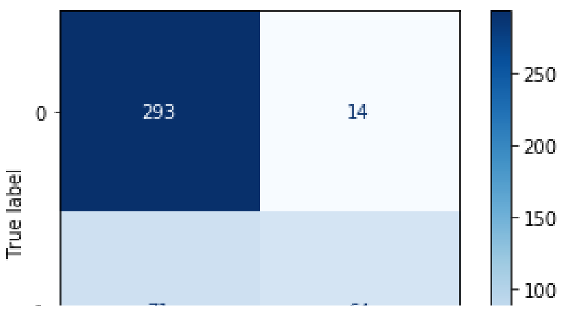


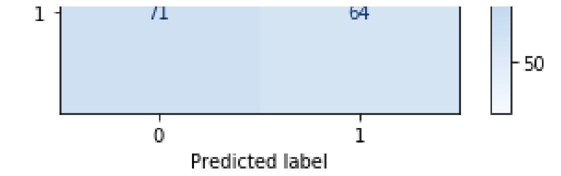

```
Train Scores:
Accuracy: 1.000
F1 Score: 1.000
ROC-AUC: 1.000
Test Scores:
Accuracy: 0.740
F1 Score: 0.566
ROC-AUC: 0.688
```

There's some serious overfitting with the training set, but that's to be expected. Again, I'm not in love with the scores on the test set. But let's keep moving. Now along with Decision Trees, we'll do a Random Forest.

In [103]:

```
rf1 = RandomForestClassifier(max_depth=7, min_samples_split=10, n_estimators=100, random
_state=0)
rf1.fit(X_train, y_train)

evaluate_model(rf1, X_train, X_test, y_train, y_test)
```





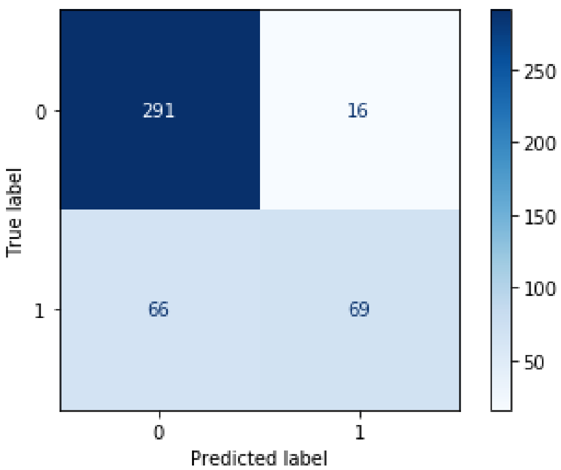

```
Train Scores:
Accuracy: 0.883
F1 Score: 0.774
ROC-AUC: 0.940
Test Scores:
Accuracy: 0.808
F1 Score: 0.601
ROC-AUC: 0.865
```

So there's a real issue with predicting the expensive bottles on this model.

Let's try it a few more times, but fiddle with the hyperparameters (that's what they're there for).

In [104]:

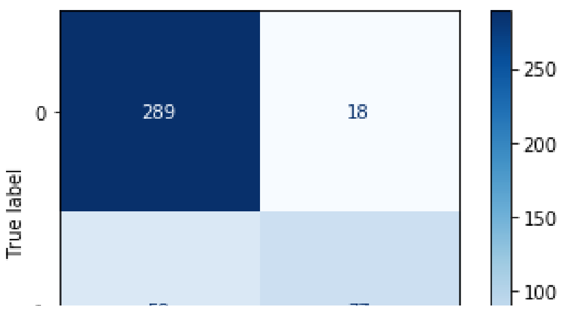
```
rf2 = RandomForestClassifier(max_depth=10, min_samples_split=15, n_estimators=100, rando
m_state=0)
rf2.fit(X_train, y_train)

evaluate_model(rf2, X_train, X_test, y_train, y_test)
```

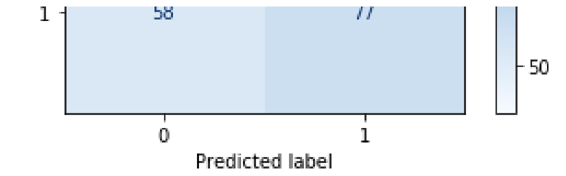



```
Train Scores:
Accuracy: 0.894
F1 Score: 0.802
ROC-AUC: 0.957
Test Scores:
Accuracy: 0.814
F1 Score: 0.627
ROC-AUC: 0.869
```

In [105]:

```
rf3 = RandomForestClassifier(max_depth=20, min_samples_split=5, n_estimators=50, random_
state=0)
rf3.fit(X_train, y_train)

evaluate_model(rf3, X_train, X_test, y_train, y_test)
```

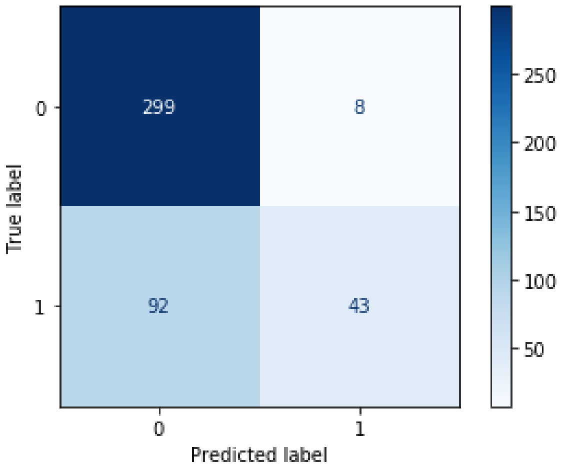





```
Train Scores:
Accuracy: 0.961
F1 Score: 0.933
ROC-AUC: 0.996
Test Scores:
Accuracy: 0.828
F1 Score: 0.670
ROC-AUC: 0.867
```

In [106]:

```
rf4 = RandomForestClassifier(max_depth=5, min_samples_split=15, n_estimators=100, random
_state=0)
rf4.fit(X_train, y_train)

evaluate_model(rf4, X_train, X_test, y_train, y_test)
```




```
Train Scores:
Accuracy: 0.832
F1 Score: 0.635
ROC-AUC: 0.913
Test Scores:
Accuracy: 0.774
F1 Score: 0.462
ROC-AUC: 0.845
```

So the deeper we go, the better the scores get, but some of these models are getting more wrong than right.

In [107]:

```
feats = rf1.feature_importances_
feature_imps = dict(zip(X.columns, feats))
feature_imps
```

Out[107]:

```
{'Meta Critic': 0.3450085902315811,
 'STDEV': 0.06867386156502786,
 '#': 0.04577546968176978,
 'Ages': 0.31489475853272314,
 'Class_Bourbon-like': 0.003053327958033772,
 'Class_Rye-like': 0.014775088379028932,
 'Class_Scotch-like': 0.003151292212340932,
 'Class_SingleMalt-like': 0.01983059600422198,
 'Cluster_A': 0.006316096259350864,
 'Cluster_B': 0.002383379650119768,
 'Cluster_C': 0.009962678234072868,
 'Cluster_E': 0.006397264428886974,
 'Cluster_F': 0.002467664788080807,
```

```
 'Cluster_G': 0.006457372473300611,
 'Cluster_H': 0.0024052317250940845,
 'Cluster_I': 0.0029902425596776113,
 'Cluster_J': 0.007719837647431484,
 'Cluster_R0': 0.0017272960333496062,
 'Cluster_R1': 0.001298766919304654,
 'Cluster_R2': 0.0007725846275398857,
 'Cluster_R3': 0.0019091030507658018,
 'Cluster_R4': 0.0015977918516986162,
 'Cluster_U': 0.02979494383372178,
 'Country_Belgium': 5.011793641205933e-05,
 'Country_Canada': 0.022701467511640056,
 'Country_England': 5.756465554997826e-05,
 'Country_Finland': 0.0007407521622841761,
 'Country_France': 9.091179654500147e-06,
 'Country_India': 0.008501102837289252,
 'Country_Ireland': 0.00235812863306727,
 'Country_Japan': 0.004406505459774526,
 'Country_Netherlands': 0.00028971153647630197,
 'Country_Scotland': 0.010676519848328896,
 'Country_South Africa': 2.5163984695654473e-05,
 'Country_Sweden': 0.009579650932509821,
 'Country_Switzerland': 0.0005960217714158389,
 'Country_Taiwan': 0.002908736008286072,
 'Country_Tasmania': 0.0013824310713002235,
 'Country_USA': 0.003637017201756514,
 'Country_Wales': 0.00022434802078561 6,
 'Type_Barley': 5.88626771862898e-07,
 'Type_Blend': 0.010207343600472154,
 'Type_Bourbon': 0.002541517744785857,
 'Type_Flavoured': 3.825998617388 4234e-05,
 'Type_Grain': 0.00014916643164694434,
 'Type_Malt': 0.01704024771760103,
 'Type_Rye': 0.002502992969315895 7,
 'Type_Wheat': 1.222674358966684e-05,
 'Type_Whiskey': 0.0}
```

In [108]:

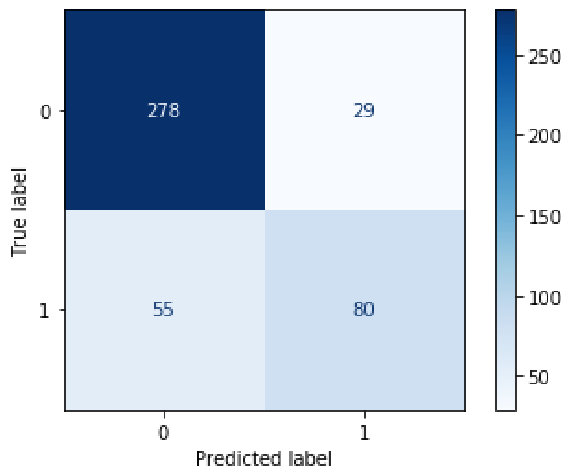
```
ada1 = AdaBoostClassifier(DecisionTreeClassifier(max_depth=5),
                          random_state=1, n_estimators=200)
ada1.fit(X_train, y_train)
ada1.score(X_test,y_test)
```

Out[108]:

```
0.8099547511312217
```

In [109]:

```
evaluate_model(ada1, X_train, X_test, y_train, y_test)
```

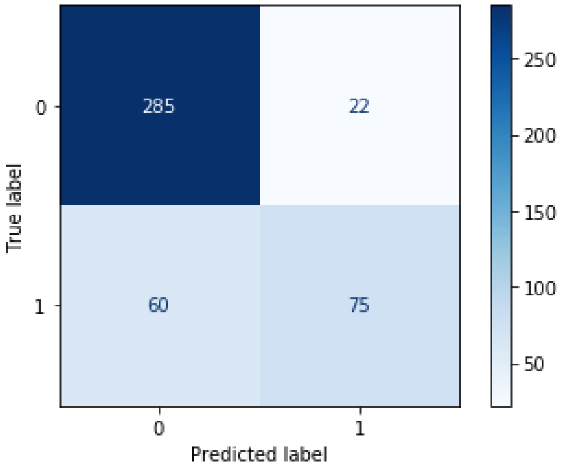



```
Train Scores:
Accuracy: 1.000
F1 Score: 1.000
ROC-AUC: 1.000
```

```
Test Scores:
Accuracy: 0.810
F1 Score: 0.656
ROC-AUC: 0.850
```

In [110]:

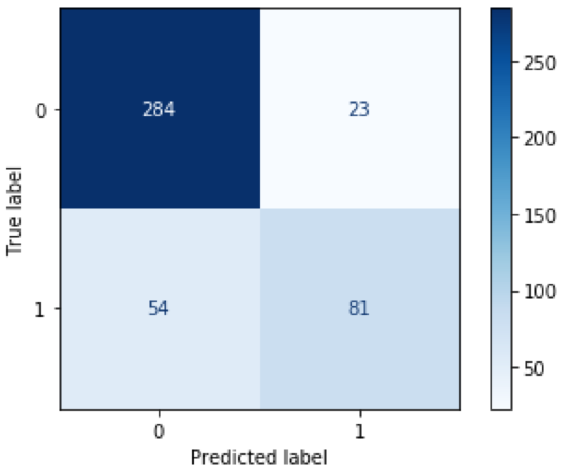
```
ada2 = AdaBoostClassifier(DecisionTreeClassifier(max_depth=10),
                          random_state=1, n_estimators=200)
ada2.fit(X_train, y_train)
evaluate_model(ada2, X_train, X_test, y_train, y_test)
```




```
Train Scores:
Accuracy: 1.000
F1 Score: 1.000
ROC-AUC: 1.000
Test Scores:
Accuracy: 0.814
F1 Score: 0.647
ROC-AUC: 0.861
```

In [111]:

```
ada3 = AdaBoostClassifier(DecisionTreeClassifier(max_depth=15),
                          random_state=1, n_estimators=200)
ada3.fit(X_train, y_train)
evaluate_model(ada3, X_train, X_test, y_train, y_test)
```




```
Train Scores:
Accuracy: 1.000
F1 Score: 1.000
ROC-AUC: 1.000
Test Scores:
Accuracy: 0.826
F1 Score: 0.678
ROC-AUC: 0.869
```

So increasing the depth on the AdaBoost didn't help the test scores at all. Weird. I guess because it's fitting so

well to the training data, it's causing more error in the test set.

Let's see how the Gradient Boosting works.

In [112]:

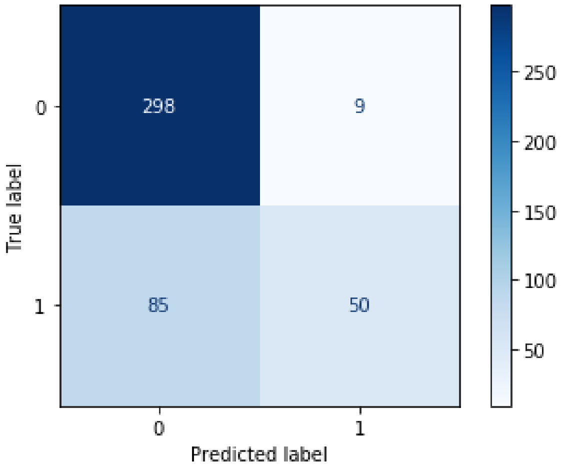
```
gbm1 = GradientBoostingClassifier(learning_rate=0.01, random_state=1)
gbm1.fit(X_train, y_train)
gbm1.score(X_test,y_test)
```

Out[112]:

```
0.7873303167420814
```

In [113]:

```
evaluate_model(gbm1, X_train, X_test, y_train, y_test)
```




```
Train Scores:
Accuracy: 0.834
F1 Score: 0.645
ROC-AUC: 0.894
Test Scores:
Accuracy: 0.787
F1 Score: 0.515
ROC-AUC: 0.848
```

In [114]:

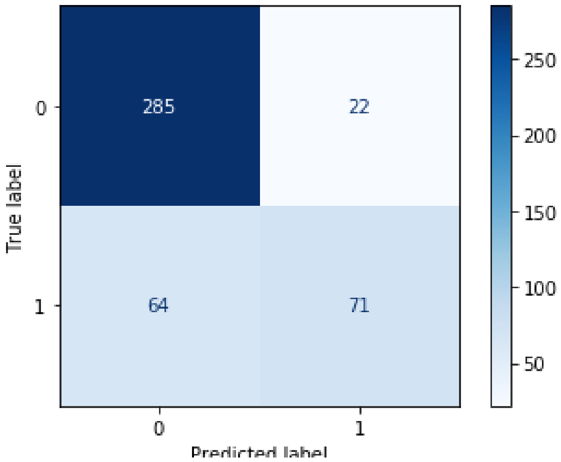
```
gbm2 = GradientBoostingClassifier(learning_rate=0.05, random_state=1)
gbm2.fit(X_train, y_train)
gbm2.score(X_test,y_test)
```

Out[114]:

```
0.8054298642533937
```

In [115]:

```
evaluate_model(gbm2, X_train, X_test, y_train, y_test)
```



```
Train Scores:
Accuracy: 0.878
F1 Score: 0.772
ROC-AUC: 0.937
Test Scores:
Accuracy: 0.805
F1 Score: 0.623
ROC-AUC: 0.877
```

In [116]:

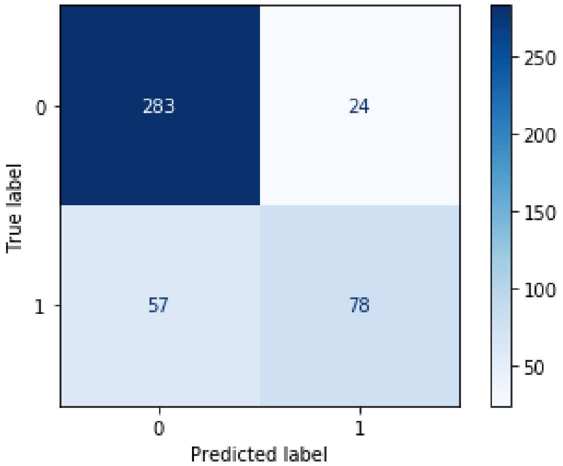
```
gbm3 = GradientBoostingClassifier(learning_rate=0.1, random_state=1)
gbm3.fit(X_train, y_train)
gbm3.score(X_test,y_test)
```

Out[116]:

```
0.8167420814479638
```

In [117]:

```
evaluate_model(gbm3, X_train, X_test, y_train, y_test)
```




```
Train Scores:
Accuracy: 0.899
F1 Score: 0.819
ROC-AUC: 0.957
Test Scores:
Accuracy: 0.817
F1 Score: 0.658
ROC-AUC: 0.885
```

So now let's try the XGBoost:

In [118]:

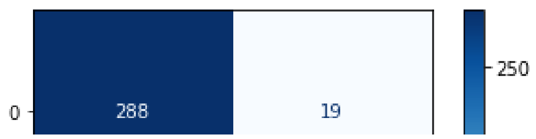
```
import xgboost as xgb
xgb1 = xgb.XGBClassifier(random_state=1, learning_rate=0.01)

xgb1.fit(X_train, y_train)
xgb1.score(X_test,y_test)
```

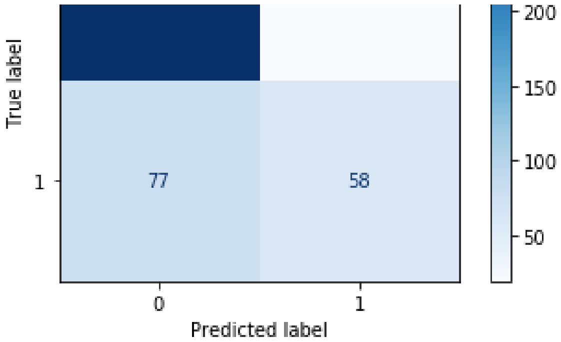
Out[118]:

```
0.7828054298642534
```

In [119]:

```
evaluate_model(xgb1, X_train, X_test, y_train, y_test)
```






```
Train Scores:
Accuracy: 0.840
F1 Score: 0.688
ROC-AUC: 0.889
Test Scores:
Accuracy: 0.783
F1 Score: 0.547
ROC-AUC: 0.843
```

In [120]:

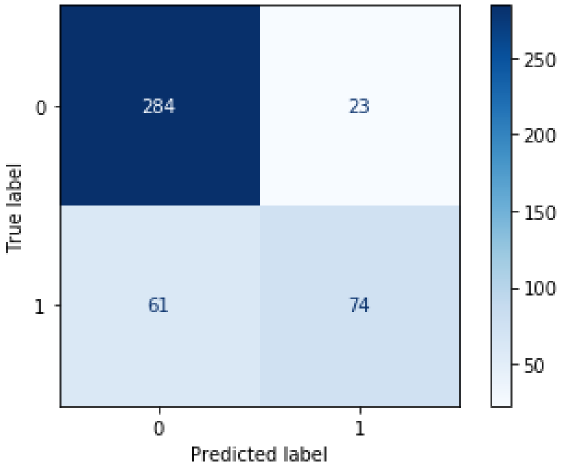
```
xgb2 = xgb.XGBClassifier(random_state=1, learning_rate=0.05)
xgb2.fit(X_train, y_train)
```

Out[120]:

```
XGBClassifier(learning_rate=0.05, random_state=1)
```

In [121]:

```
evaluate_model(xgb2, X_train, X_test, y_train, y_test)
```




```
Train Scores:
Accuracy: 0.865
F1 Score: 0.751
ROC-AUC: 0.928
Test Scores:
Accuracy: 0.810
F1 Score: 0.638
ROC-AUC: 0.873
```

In [122]:

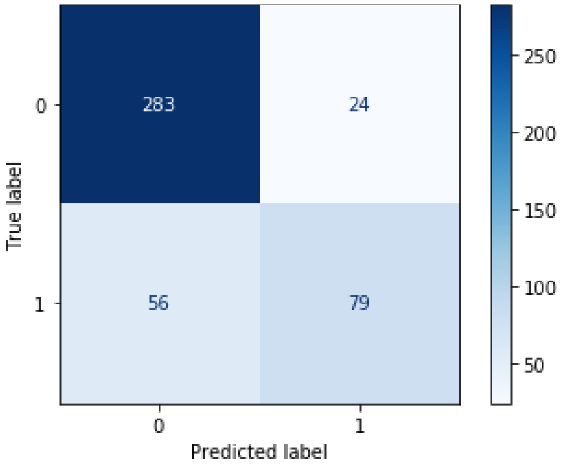
```
xgb3 = xgb.XGBClassifier(random_state=1, learning_rate=0.1)
xgb3.fit(X_train, y_train)
```

Out[122]:

```
XGBClassifier(random_state=1)
```

In [123]:

```
evaluate_model(xgb3, X_train, X_test, y_train, y_test)
```




```
Train Scores:
Accuracy: 0.884
F1 Score: 0.790
ROC-AUC: 0.946
Test Scores:
Accuracy: 0.819
F1 Score: 0.664
ROC-AUC: 0.885
```

In [124]:

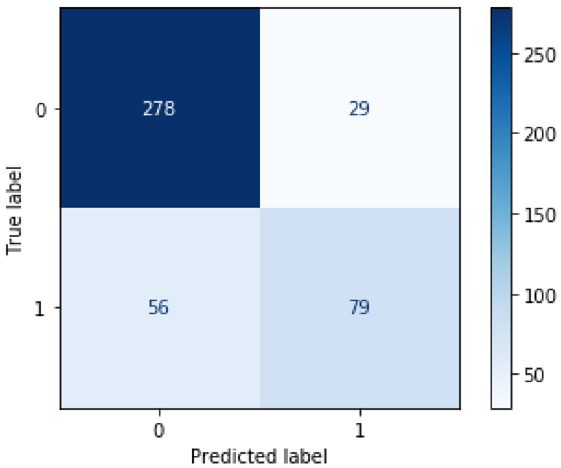
```
clf = DecisionTreeClassifier()

param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [1, 2, 5, 10],
    'min_samples_split': [3, 5, 10, 20]
}

gs_tree = GridSearchCV(clf, param_grid, cv=3)
gs_tree.fit(X_train, y_train)

gs_testing_score = gs_tree.score(X_test, y_test)
gs_tree.best_params_
```

Out[124]:

```
{'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 3}
```

In [125]:

```
gs_testing_score
evaluate_model(gs_tree, X_train, X_test, y_train, y_test)
```



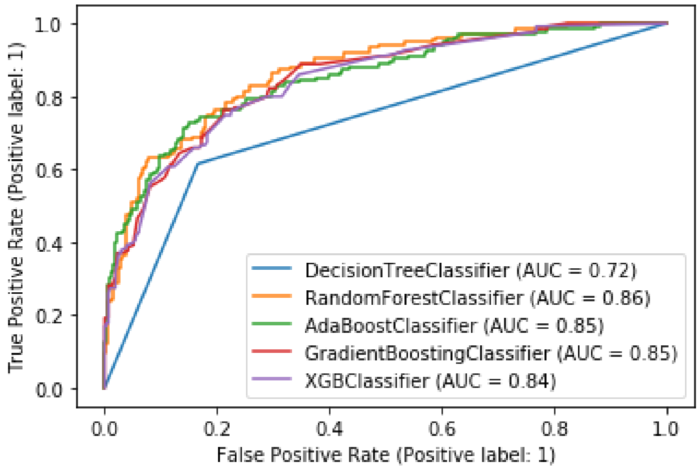

```
Train Scores:
Accuracy: 0.850
F1 Score: 0.739
ROC-AUC: 0.903
Test Scores:
Accuracy: 0.808
```

```
F1 Score: 0.650
ROC-AUC: 0.805
```

Let's look at these, and see how they compare.

In [126]:

```
models = [tree, rf1, ada1, gbm1, xgb1]

fig, ax = plt.subplots()

for model in models:
    plot_roc_curve(model, X_test, y_test, ax=ax)
```



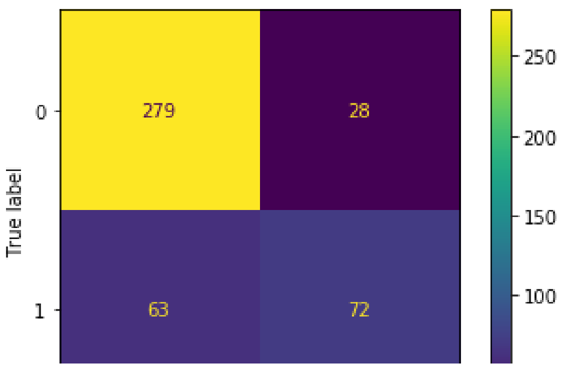
In [127]:

```
svc_lin1 = SVC(kernel='linear', C=1)
svc_lin1.fit(X_train, y_train)

y_pred_train = svc_lin1.predict(X_train)
y_pred_test = svc_lin1.predict(X_test)
```

In [128]:

```
print(classification_report(y_test, y_pred_test))
print(f"Train accuracy: {accuracy_score(y_train, y_pred_train):.4f}")
print(f"Test accuracy: {accuracy_score(y_test, y_pred_test):.4f}")

plot_confusion_matrix(svc_lin1, X_test, y_test)
plt.show()
```

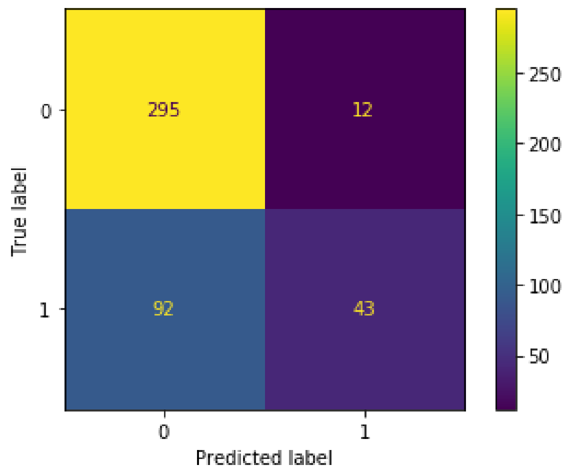
```
              precision    recall  f1-score   support

           0       0.82      0.91      0.86       307
           1       0.72      0.53      0.61       135

    accuracy                           0.79       442
   macro avg       0.77      0.72      0.74       442
weighted avg       0.79      0.79      0.78       442

Train accuracy: 0.8466
Test accuracy: 0.7941
```



In [129]:

```
svc_rbf1 = SVC(kernel='rbf', C=1, gamma='scale')
svc_rbf1.fit(X_train, y_train)

y_pred_train = svc_rbf1.predict(X_train)
y_pred_test = svc_rbf1.predict(X_test)
print(classification_report(y_test, y_pred_test))
print(f"Train accuracy: {accuracy_score(y_train, y_pred_train):.4f}")
print(f"Test accuracy: {accuracy_score(y_test, y_pred_test):.4f}")

plot_confusion_matrix(svc_rbf1, X_test, y_test)
plt.show()
```

```
              precision    recall  f1-score   support

           0       0.76      0.96      0.85       307
           1       0.78      0.32      0.45       135

    accuracy                           0.76       442
   macro avg       0.77      0.64      0.65       442
weighted avg       0.77      0.76      0.73       442

Train accuracy: 0.7952
Test accuracy: 0.7647
```

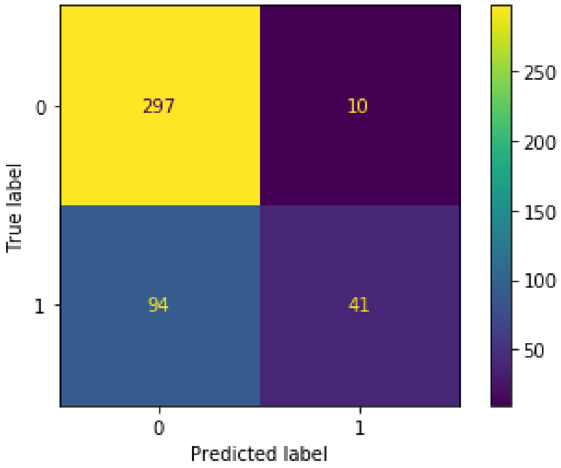


In [130]:

```
svc_poly1 = SVC(kernel='poly', C=1, gamma='scale', degree=3) # using mostly default valu
es here
svc_poly1.fit(X_train, y_train)

y_pred_train = svc_poly1.predict(X_train)
y_pred_test = svc_poly1.predict(X_test)
print(classification_report(y_test, y_pred_test))
print(f"Train accuracy: {accuracy_score(y_train, y_pred_train):.4f}")
print(f"Test accuracy: {accuracy_score(y_test, y_pred_test):.4f}")

plot_confusion_matrix(svc_poly1, X_test, y_test)
plt.show()
```

```
              precision    recall  f1-score   support

           0       0.76      0.97      0.85       307
           1       0.80      0.30      0.44       135

    accuracy                           0.76       442
   macro avg       0.78      0.64      0.65       442
weighted avg       0.77      0.76      0.73       442

Train accuracy: 0.8005
```

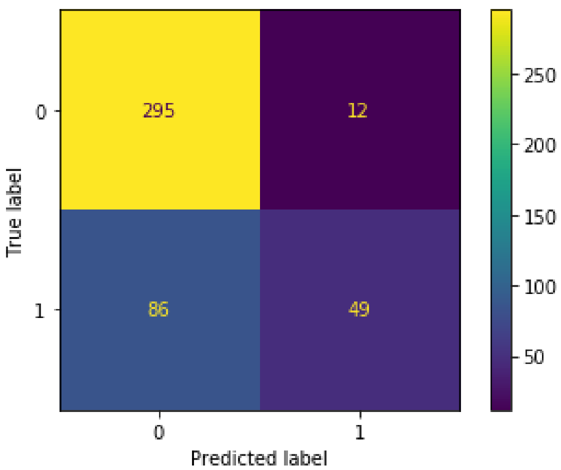
```
Test accuracy: 0.7647
```



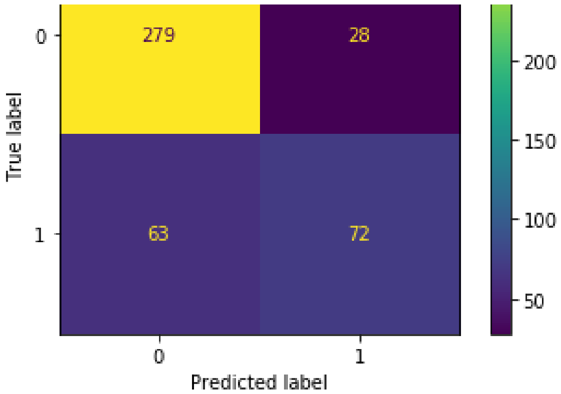
It looks like the clear winner of the SVM is the linear kernel at least it has more correct True predictions than incorrect. So let's use linear and try different C values.

In [131]:

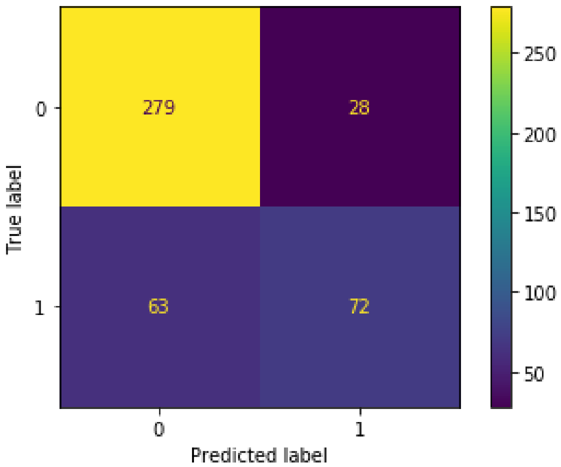
```python
for c in [.01,1,10,100]:
    svc_c = SVC(kernel='linear', C=c, gamma='scale') # going linear again
    svc_c.fit(X_train, y_train)

    y_pred_train = svc_c.predict(X_train)
    y_pred_test = svc_c.predict(X_test)
    plot_confusion_matrix(svc_c, X_test, y_test)
    plt.show()

    print("-----")
    print(f'Results at C = {c}')
    print(classification_report(y_test, y_pred_test))
    print(f"Train accuracy: {accuracy_score(y_train, y_pred_train):.4f}")
    print(f"Test accuracy: {accuracy_score(y_test, y_pred_test):.4f}")
```



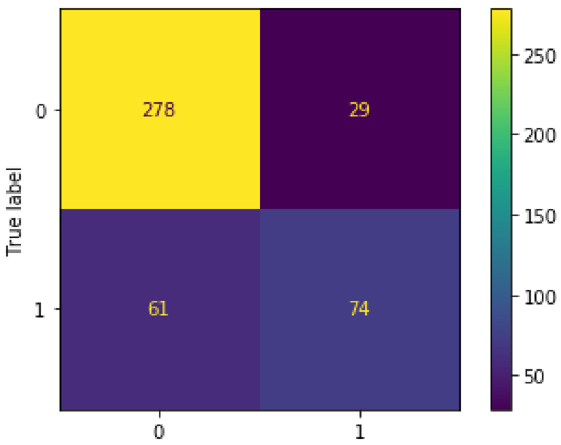

```
-----
Results at C = 0.01
              precision    recall  f1-score   support

           0       0.77      0.96      0.86       307
           1       0.80      0.36      0.50       135

    accuracy                           0.78       442
   macro avg       0.79      0.66      0.68       442
weighted avg       0.78      0.78      0.75       442

Train accuracy: 0.8156
Test accuracy: 0.7783
```




```
-----
Results at C = 1
              precision    recall  f1-score   support

           0       0.82      0.91      0.86       307
           1       0.72      0.53      0.61       135

    accuracy                           0.79       442
   macro avg       0.77      0.72      0.74       442
weighted avg       0.79      0.79      0.78       442

Train accuracy: 0.8466
Test accuracy: 0.7941
```




```
-----
Results at C = 10
              precision    recall  f1-score   support

           0       0.82      0.91      0.86       307
           1       0.72      0.53      0.61       135

    accuracy                           0.79       442
   macro avg       0.77      0.72      0.74       442
weighted avg       0.79      0.79      0.78       442

Train accuracy: 0.8534
Test accuracy: 0.7941
```



Predicted label

```
-----
Results at C = 100
              precision    recall  f1-score   support

           0       0.82      0.91      0.86       307
           1       0.72      0.55      0.62       135

    accuracy                           0.80       442
   macro avg       0.77      0.73      0.74       442
weighted avg       0.79      0.80      0.79       442

Train accuracy: 0.8541
Test accuracy: 0.7964
```

It looks like we're reaching an upper limit of .85/.79 with C=10, anything above that seems to not matter. So just to try something else, I want to see what scores we get when we don't use the meta critic data.

In [132]:

```
X.head()
```

Out[132]:

	Meta Critic	STDEV	#	Ages	Class_Bourbon-like	Class_Rye-like	Class_Scotch-like	Class_SingleMalt-like	Cluster_A	Cluster_B	...	Country_
0	9.57	0.24	3	10	0	0	0	1	1	0	...	
1	9.48	0.23	3	42	0	0	0	1	0	0	...	
2	9.42	0.23	4	27	0	0	0	1	0	0	...	
3	9.29	0.26	17	40	0	0	0	1	1	0	...	
4	9.24	0.22	21	25	0	0	0	1	1	0	...	

5 rows × 49 columns

In [133]:

```
drops=['Meta Critic','STDEV','#']
```

In [134]:

```
X2=X.drop(drops, axis=1)
```

In [135]:

```
X2.head()
```

Out[135]:

	Ages	Class_Bourbon-like	Class_Rye-like	Class_Scotch-like	Class_SingleMalt-like	Cluster_A	Cluster_B	Cluster_C	Cluster_E	Cluster_F
0	10	0	0	0	1	1	0	0	0	0
1	42	0	0	0	1	0	0	1	0	0
2	27	0	0	0	1	0	0	1	0	0
3	40	0	0	0	1	1	0	0	0	0
4	25	0	0	0	1	1	0	0	0	0

5 rows × 46 columns

In [136]:

```
X_train,X_test,y_train,y_test=train_test_split(X2,y,test_size= 0.25, random_state=42)
```

In [137]:

```
model_log2 = logreg.fit(X_train, y_train).decision_function(X_test)
```

In [138]:

```
y_hat_train = logreg.predict(X_train)
y_hat_test = logreg.predict(X_test)
cnf_matrix = confusion_matrix(y_test, y_hat_test)
print('Confusion Matrix:\n', cnf_matrix)
```

```
Confusion Matrix:
 [[275  32]
 [ 75  60]]
```

In [139]:

```
residuals = np.abs(y_test - y_hat_test)
print(pd.Series(residuals).value_counts())
print('------------------------------------')
print(pd.Series(residuals).value_counts(normalize=True))
```

```
0    335
1    107
dtype: int64
------------------------------------
0    0.757919
1    0.242081
dtype: float64
```

So this model is slightly worse. Let's see how the best performing tree model does.

In [140]:

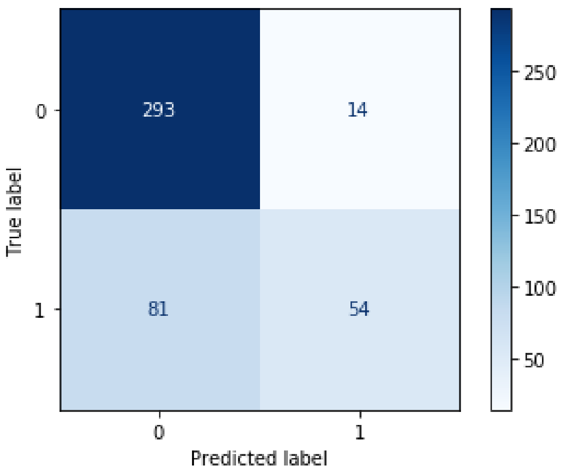
```
tree.fit(X_train, y_train)
```

Out[140]:

```
DecisionTreeClassifier()
```

In [141]:

```
evaluate_model(tree, X_train, X_test, y_train, y_test)
```




```
Train Scores:
Accuracy: 0.868
F1 Score: 0.736
ROC-AUC: 0.914
Test Scores:
Accuracy: 0.785
F1 Score: 0.532
ROC-AUC: 0.803
```

In [142]:

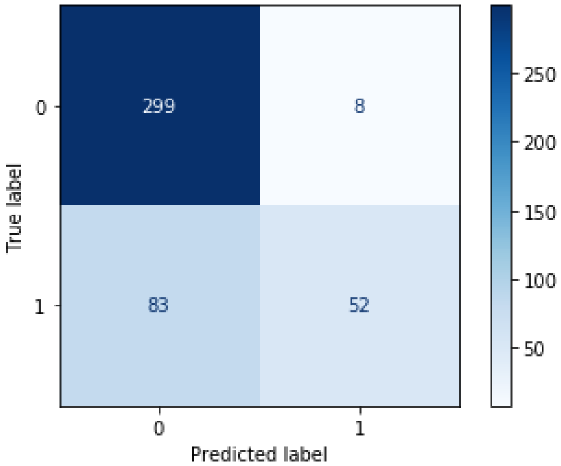
```
xgb3.fit(X_train, y_train)
```

Out[142]:

```
XGBClassifier(random_state=1)
```

In [143]:

```
evaluate_model(xgb3, X_train, X_test, y_train, y_test)
```

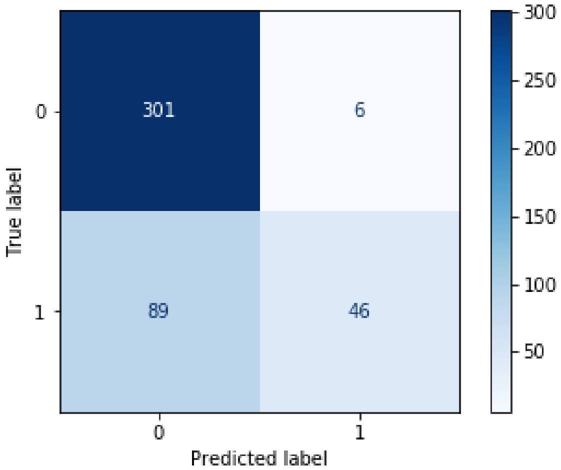



```
Train Scores:
Accuracy: 0.840
F1 Score: 0.666
ROC-AUC: 0.873
Test Scores:
Accuracy: 0.794
F1 Score: 0.533
ROC-AUC: 0.796
```

So it looks like eliminating those columns made the model worse.

In [144]:

```
tree4=DecisionTreeClassifier(criterion='entropy',max_depth=5,min_samples_split=10)
tree4.fit(X_train,y_train)
evaluate_model(tree4, X_train, X_test, y_train, y_test)
```




```
Train Scores:
Accuracy: 0.813
F1 Score: 0.581
ROC-AUC: 0.779
Test Scores:
Accuracy: 0.785
F1 Score: 0.492
ROC-AUC: 0.735
```

So this was one of the best performing models originally, and now the values in the confusion matrix are completely backwards. So it turns out the other columns were needed. I'll put them back here shortly.

In [145]:

```
test_pred=tree4.predict(X_test)
```

In [146]:

```
test_pred
```

Out[146]:

```
array([0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 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, 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,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 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, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0], dtype=int64)
```

In [147]:

```
y_test
```

Out[147]:

```
1559    0
212     0
1682    0
836     0
1245    0
       ..
1624    0
1755    0
1318    0
1193    0
985     0
Length: 442, dtype: int64
```

In [148]:

```
test_df=pd.DataFrame(y_test, columns=['actual'])
```

In [149]:

```
test_df['pred']=test_pred
```

In [150]:

```
test_df.loc[(test_df.actual==0)&(test_df.pred==1)]
```

Out[150]:

	actual	pred
251	0	1

	actual	pred
65	0	1
755	0	1
49	0	1
289	0	1
543	0	1

In [151]:

```
X_test
```

Out[151]:

	Ages	Class_Bourbon-like	Class_Rye-like	Class_Scotch-like	Class_SingleMalt-like	Cluster_A	Cluster_B	Cluster_C	Cluster_E	Cluste
1561	2	1	0	0	0	0	0	0	0	
212	15	0	0	0	1	0	0	1	0	
1684	3	1	0	0	0	0	0	0	0	
837	12	0	0	0	1	0	0	0	0	
1247	3	0	0	1	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
1626	2	1	0	0	0	0	0	0	0	
1757	2	1	0	0	0	0	0	0	0	
1320	2	0	1	0	0	0	0	0	0	
1195	18	0	0	1	0	0	0	0	0	
986	3	0	0	0	1	0	0	0	0	

442 rows × 46 columns

So every model I've run has had some incorrect guesses. Perhaps there's a way to use these, as recommendations: whiskies that the model thinks are expensive ("should be" expensive) but are actually cheap.

In [152]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size= 0.25, random_state=42)
```

In [153]:

```
tree=DecisionTreeClassifier(criterion='entropy',max_depth=5,min_samples_split=10)
```

In [154]:

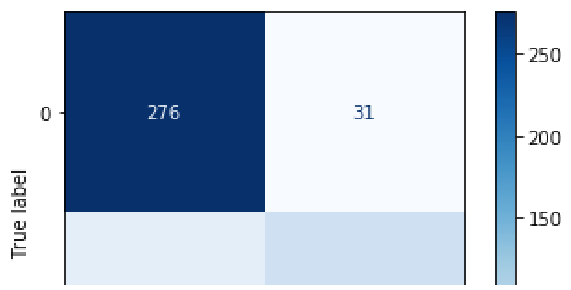
```
tree.fit(X_train,y_train)
```

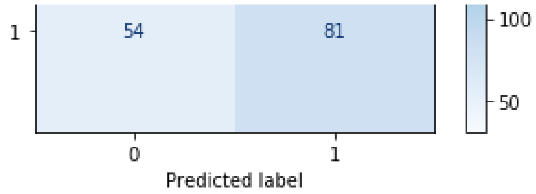
Out[154]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=5, min_samples_split=10)
```

In [155]:

```
evaluate_model(tree, X_train, X_test, y_train, y_test)
```






```
Train Scores:
Accuracy: 0.847
F1 Score: 0.734
ROC-AUC: 0.898
Test Scores:
Accuracy: 0.808
F1 Score: 0.656
ROC-AUC: 0.814
```

In [156]:

```
tree_test_pred=tree.predict(X_test)
```

In [157]:

```
tree_test_df=pd.DataFrame(y_test, columns=['actual'])
```

In [158]:

```
tree_test_df['pred']=tree_test_pred
```

In [159]:

```
tree_test_df
```

Out[159]:

	actual	pred
1559	0	1
212	0	0
1682	0	0
836	0	0
1245	0	0
...	...	...
1624	0	0
1755	0	0
1318	0	0
1193	0	0
985	0	0

442 rows × 2 columns

In [160]:

```
act_cheap=tree_test_df.loc[(tree_test_df.actual==0)&(tree_test_df.pred==1)]
```

In [161]:

```
act_cheap
```

Out[161]:

	actual	pred
1559	0	1
251	0	1

	actual	pred
65	0	1
901	0	1
1196	0	1
1581	0	1
1357	0	1
755	0	1
49	0	1
1569	0	1
76	0	1
1365	0	1
99	0	1
1323	0	1
1614	0	1
1360	0	1
289	0	1
433	0	1
51	0	1
1563	0	1
1087	0	1
1055	0	1
1587	0	1
1377	0	1
1052	0	1
1125	0	1
543	0	1
1182	0	1
1083	0	1
1342	0	1
1604	0	1

I have to go back to df1 for the names. So I can iterate over the index in my cheap dataframe and grab some names.

In [162]:

```
df1.head()
```

Out[162]:

	Whisky	Meta Critic	STDEV	#	Class	Cluster	Country	Type
0	Macallan 10yo Full Proof 57% 1980 (OB, Giovine...	9.57	0.24	3	SingleMalt-like	A	Scotland	Malt
1	Ledaig 42yo Dusgadh	9.48	0.23	3	SingleMalt-like	C	Scotland	Malt
2	Laphroaig 27yo 57.4% 1980-2007 (OB, 5 Oloroso ...	9.42	0.23	4	SingleMalt-like	C	Scotland	Malt
3	Glenfarclas 40yo	9.29	0.26	17	SingleMalt-like	A	Scotland	Malt
4	Glengoyne 25yo	9.24	0.22	21	SingleMalt-like	A	Scotland	Malt

In [163]:

```
act_cheap['name']='Unknown'
```

```
D:\anaconda3\envs\learn-env\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

In [164]:

```
act_cheap
```

Out[164]:

	actual	pred	name
1559	0	1	Unknown
251	0	1	Unknown
65	0	1	Unknown
901	0	1	Unknown
1196	0	1	Unknown
1581	0	1	Unknown
1357	0	1	Unknown
755	0	1	Unknown
49	0	1	Unknown
1569	0	1	Unknown
76	0	1	Unknown
1365	0	1	Unknown
99	0	1	Unknown
1323	0	1	Unknown
1614	0	1	Unknown
1360	0	1	Unknown
289	0	1	Unknown
433	0	1	Unknown
51	0	1	Unknown
1563	0	1	Unknown
1087	0	1	Unknown
1055	0	1	Unknown
1587	0	1	Unknown
1377	0	1	Unknown
1052	0	1	Unknown
1125	0	1	Unknown
543	0	1	Unknown
1182	0	1	Unknown
1083	0	1	Unknown
1342	0	1	Unknown
1604	0	1	Unknown

In [165]:

```
for i in act_cheap['name'].index:
```

```
    act_cheap.at[i,'name']=df1.at[i,'Whisky']
```

In [166]:

```
act_cheap.head(10)
```

Out[166]:

	actual	pred	name
1559	0	1	Parker's Heritage 5th 10yo Cognac Barrel Finished
251	0	1	Aberfeldy 18yo
65	0	1	BenRiach 17yo Solstice 2nd Peated Port Finish
901	0	1	Bruichladdich Port Charlotte PC10 Tro Na Linntean
1196	0	1	Whyte & Mackay 30yo
1581	0	1	Willett Family Estate 17yo Bourbon
1357	0	1	J.P. Wiser's Union 52
755	0	1	Glen Grant 10yo (G&M)
49	0	1	BenRiach 17yo Solstice Peated Port (both editi...
1569	0	1	Stagg Jr batch 5 (129.7 proof)

Conclusion

There are a few models here that have some serious potential and could really work in predicting the expensiveness of a bottle of whisky. All in all, I'm pretty satisfied with how this turned out. I think there's still a lot to play with and explore to get the models better, perhaps some research to eliminate the unknown cluster.