Beer Early BPOM Prediction

MEGA KRISTIANI

Purwadhika - Data Science - Batch 2

Biodata

Nama : Mega Kristiani
Tempat, Tanggal Lahir : Cimahi, 29 Juni 1995

Pendidikan:
Diploma III Analis Kesehatan Politeknik TEDC Bandung (2015)
Data Science - Purwadhika Bandung (2020)

Pengalaman Kerja:
Medical Representative PT. Dexa Medica (Desember 2016 - Oktober 2017)
PT. Gucci Ratu Textile Industry (November 2017 - Februari 2019

0813 2414 1813 | sheismega@gmail.com

Latar Belakang

Beer merupakan salah satu jenis minuman beralkohol yang dibuat melalui proses fermentasi, dan merupakan minuman berakohol tertua, juga paling banyak dikonsumsi di dunia. Di Indonesia, angka konsumsi beer hanya berkisar 0.6 liter/tahun (thejakartapost.com, 2018). Walaupun begitu, Indonesia memproduksi beberapa merk beer (Bintang, Stark Craft Beer, Anker Beer, Bali Hai) dan juga memiliki beer tradisional, yakni Bir Pletok, Bir Kotjok, Bir, Jawa, dan masih banyak lainnya. Salah satu yang menjadi faktor industri beer indonesia tidak berkembang adalah persyaratan BPOM yang mewajibkan sertifikasi produk sebelum beer dijual. Final Project ini akan memprediksi apakah beer yang dibuat akan lolos sertifikasi atau tidak.

Highlights

1. Membuat prediksi apakah Beer akan mendapatkan sertifikat BPOM atau tidak.
2. Membantu perusahaan beer untuk lebih produktif dan berkembang.
3. Mempermudah BPOM untuk membuat keputusan.

Dataset

Sumber: https://www.kaggle.com/jtrofe/beer-recipes

Kolom:

23 Kolom

+4 Kategorikal, 19 Non Kategorikal

Data Frame tambahan:

1 Data Frame tambahan dan 1 kolom dari refrensi.

sumber: https://www.kaggle.com/jtrofe/beer-recipes dan BPOM

	Feature	Type	Null	NullPct	Unique	Sample
0	BeerID	int64	0	0.000000	73861	[54373, 59643]
1	Name	object	1	0.000014	59148	[XMA 1.1, Premium Lager Heineken]
2	URL	object	0	0.000000	73861	[/homebrew/recipe/view/515438/left-overs-ipa, ...
3	Style	object	596	0.008069	175	[Premium American Lager, Experimental Beer]
4	StyleID	int64	0	0.000000	176	[41, 116]
5	Size(L)	float64	0	0.000000	1065	[4.3, 416.4]
6	OG	float64	0	0.000000	2036	[12.2847, 9.07809]
7	FG	float64	0	0.000000	1958	[2.91457, 4.11471]
8	ABV	float64	0	0.000000	1502	[15.59, 11.27]
9	IBU	float64	0	0.000000	12587	[73.89, 94.54]
10	Color	float64	0	0.000000	4729	[11.16, 39.64]
11	BoilSize	float64	0	0.000000	1973	[255.0, 21.12]
12	BoilTime	int64	0	0.000000	75	[84, 99]
13	BoilGravity	float64	2990	0.040481	509	[13.6, 1.0170000000000001]
14	Efficiency	float64	0	0.000000	272	[61.2, 53.6]
15	MashThickness	float64	29864	0.404327	567	[0.87, 2.85]
16	SugarScale	object	0	0.000000	2	[Plato, Specific Gravity]
17	BrewMethod	object	0	0.000000	4	[All Grain, extract]
18	PitchRate	float64	39252	0.531431	9	[2.0, 1.25]
19	PrimaryTemp	float64	22662	0.306820	217	[26.4, -6.67]
20	PrimingMethod	object	67095	0.908395	874	[Dry Hop, maple syrup]
21	PrimingAmount	object	69087	0.935365	1896	[138, 2 tabs]
22	UserId	float64	50490	0.683581	2785	[93607.0, 69219.0]

	count	mean	std	min	25%	50%	75%	max
BeerID	73861.0	36931.000000	21321.978453	1.000	18466.000	36931.000	55396.000	73861.0000
StyleID	73861.0	60.179432	56.811462	1.000	10.000	35.000	111.000	176.0000
Size(L)	73861.0	43.929775	180.373492	1.000	18.930	20.820	23.660	9200.0000
OG	73861.0	1.406266	2.196908	1.000	1.051	1.058	1.069	34.0345
FG	73861.0	1.075865	0.432524	-0.003	1.011	1.013	1.017	23.4246
ABV	73861.0	6.136865	1.883510	0.000	5.080	5.790	6.830	54.7200
IBU	73861.0	44.276186	42.945508	0.000	23.370	35.770	56.380	3409.3000
Color	73861.0	13.404989	11.944511	0.000	5.170	8.440	16.790	186.0000
BoilSize	73861.0	49.724919	193.246427	1.000	20.820	27.440	30.000	9700.0000
BoilTime	73861.0	65.074870	15.024228	0.000	60.000	60.000	60.000	240.0000
BoilGravity	70871.0	1.353955	1.930989	0.000	1.040	1.047	1.060	52.6000
Efficiency	73861.0	66.354881	14.091686	0.000	65.000	70.000	75.000	100.0000
MashThickness	43997.0	2.127235	1.682347	0.000	1.500	1.500	3.000	100.0000
PitchRate	34609.0	0.750468	0.394262	0.000	0.350	0.750	1.000	2.0000
PrimaryTemp	51199.0	19.175641	4.219676	-17.780	18.000	20.000	20.000	114.0000
UserId	23371.0	43078.069188	27734.252556	49.000	20984.000	42897.000	57841.000	134362.0000

Nan Value

	Feature	Type	Null	NullPct	Unique	Sample
1	Name	object	1	0.000014	59148	[XMA 1.1, Premium Lager Heineken]
3	Style	object	596	0.008069	175	[Premium American Lager, Experimental Beer]
13	BoilGravity	float64	2990	0.040481	509	[13.6, 1.0170000000000001]
15	MashThickness	float64	29864	0.404327	567	[0.87, 2.85]
18	PitchRate	float64	39252	0.531431	9	[2.0, 1.25]
19	PrimaryTemp	float64	22662	0.306820	217	[26.4, -6.67]
20	PrimingMethod	object	67095	0.908395	874	[Dry Hop, maple syrup]
21	PrimingAmount	object	69087	0.935365	1896	[138, 2 tabs]
22	UserId	float64	50490	0.683581	2785	[93607.0, 69219.0]

Data Preprocessing

1. Handling Nan Value

	Feature	Type	Null	NullPct	Unique	Sample
🗑	Name	object	1	0.000014	59148	[XMA 1.1, Premium Lager Heineken]
🗑	Style	object	596	0.008069	175	[Premium American Lager, Experimental Beer]
13	BoilGravity	float64	2990	0.040481	509	[13.6, 1.0170000000000001]
15	MashThickness	float64	29864	0.404327	567	[0.87, 2.85]
🗑	PitchRate	float64	39252	0.531431	9	[2.0, 1.25]
19	PrimaryTemp	float64	22662	0.306820	217	[26.4, -6.67]
🗑	PrimingMethod	object	67095	0.908395	874	[Dry Hop, maple syrup]
🗑	PrimingAmount	object	69087	0.935365	1896	[138, 2 tabs]
🗑	UserId	float64	50490	0.683581	2785	[93607.0, 69219.0]

🗑 = drop column atau row

Name

```
name=df[df['Name'].isnull()]
name
```

	BeerID	Name	UR
28051	28052	NaN	/homebrew/recipe/view/396948/

```
df.shape
```

```
(73861, 19)
```

```
df.drop(name.index, axis=0, inplace=True)
df.shape
```

```
(73860, 19)
```

Style

```
In [17]: style['StyleID'].value_counts()
```

```
Out[17]: 111    596
         Name: StyleID, dtype: int64
```

```
In [18]: df2[df2['StyleID']==111]
```

Out[18]:

	Style	StyleID
110	NaN	111

```
In [19]: df.shape
```

```
Out[19]: (73860, 19)
```

```
In [20]: df.dropna(subset=['Style'], axis=0, inplace=True)
         df.shape
```

```
Out[20]: (73264, 19)
```

Boil Gravity

```
kor['BoilGravity'].sort_values(ascending=False)[1:]
```

```
OG               0.968138
FG               0.908611
Size(L)          0.176071
BoilSize         0.175972
Efficiency       0.059132
MashThickness    0.050944
BoilTime         0.026921
ABV              0.025421
PitchRate        0.016216
Color            0.013451
StyleID          0.002326
PrimaryTemp      0.002268
UserId          -0.003206
IBU             -0.007498
BeerID          -0.031692
Name: BoilGravity, dtype: float64
```

```
nan['Feature'].values
```

```
array(['Name', 'Style', 'BoilGravity', 'MashThickness', 'PitchRate',
       'PrimaryTemp', 'PrimingMethod', 'PrimingAmount', 'UserId'],
      dtype=object)
```

```
# Making new dataframe
dfbg=df[['OG','FG','Size(L)','BoilSize','BoilGravity']]
dfbg.dropna(subset=['BoilGravity'], axis=0, inplace=True)
dfbg.head()
```

Boil Gravity

```
from sklearn import metrics
print('Data Test:')
print('MAE:', metrics.mean_absolute_error(y_test, pred_test))
print('MSE:', metrics.mean_squared_error(y_test, pred_test))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred_test)))
print('R2 Score:', metrics.r2_score(y_test,pred_test))
```

```
Data Test:
MAE: 0.04452177439694495
MSE: 0.15793494550209325
RMSE: 0.39741029868650013
R2 Score: 0.956862676533781
```

```
from sklearn import metrics
print('Data Test:')
print('MAE:', metrics.mean_absolute_error(y_train, pred_train))
print('MSE:', metrics.mean_squared_error(y_train, pred_train))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_train, pred_train)))
print('R2 Score:', metrics.r2_score(y_train,pred_train))
```

```
Data Test:
MAE: 0.05159950878114774
MSE: 0.24487385179737634
RMSE: 0.49484730149549805
R2 Score: 0.9342130511529688
```

Mash Thickness

```
# fill nan value Mash Thickness
df['MashThickness']=df.groupby(['StyleID','SugarScale'])['MashThickness'].apply(lambda x: x.fillna(x.mean()))
df['MashThickness']=df['MashThickness'].fillna(df['MashThickness'].mean())
df[df['MashThickness'].isnull()]
```

BeerID	Name	URL	Style	StyleID	Size(L)	OG	FG	ABV	IBU	Color	BoilSize	BoilTime	BoilGravity	Efficiency	MashThickness	SugarScale	BrewMethod

Primary Temp

```
from statistics import mode
df['PrimaryTemp']=df.groupby(['StyleID','SugarScale'])['PrimaryTemp'].apply(lambda x: x.fillna(x.mode()))
df['PrimaryTemp']=df['MashThickness'].fillna(df['PrimaryTemp'].mode())
df.head()
```

2. Droping Non Value Features

URL Column

```
# dropping URL column
df.drop('URL', axis=1, inplace=True)
df.head()
```

	Name	Style	StyleID	Size(L)	OG	FG	ABV	IBU	Color	BoilSize	BoilTime	BoilGravity	Efficiency	MashThickness	SugarScale	BrewM
0	Vanilla Cream Ale	Cream Ale	45	21.77	1.055	1.013	5.48	17.65	4.83	28.39	75	1.038000	70.0	1.905722	Specific Gravity	All

BeerID

```
df.drop('BeerID', axis=1, inplace=True)
df.columns
```

```
Index(['Name', 'Style', 'StyleID', 'Size(L)', 'OG', 'FG', 'ABV', 'IBU',
       'Color', 'BoilSize', 'BoilTime', 'BoilGravity', 'Efficiency',
       'MashThickness', 'SugarScale', 'BrewMethod', 'PrimaryTemp'],
      dtype='object')
```

3. Adding New Feature

BPOM

Syarat BPOM:

1. Beer: Kadar alkohol <8%
2. Stout: Kadar alkohol 3-8%

```
df['BPOM']=0
```

```
beer=df[df['Color']<24]
idxb=beer.index
stout=df[df['Color']>=24]
idxs=stout.index
```

```
a=beer[beer['ABV']<2]
idxa=a.index
b=beer[(beer['ABV']>=2) & (beer['ABV']<=8)]
idxb=b.index
c=beer[beer['ABV']>8]
idxc=c.index
```

```
d=stout[stout['ABV']<2]
idxd=d.index
e=stout[(stout['ABV']>=2) & (stout['ABV']<=8)]
idxe=e.index
f=stout[stout['ABV']>8]
idxf=f.index
```

```
df.loc[idxa,'BPOM']=1
df.loc[idxb,'BPOM']=2
df.loc[idxc,'BPOM']=3
df.loc[idxd,'BPOM']=4
df.loc[idxe,'BPOM']=5
df.loc[idxf,'BPOM']=6
```

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

```
2    55165
5     9372
3     5503
6     2963
1      250
4       11
Name: BPOM, dtype: int64
```

BPOM
2
2
2
2
2

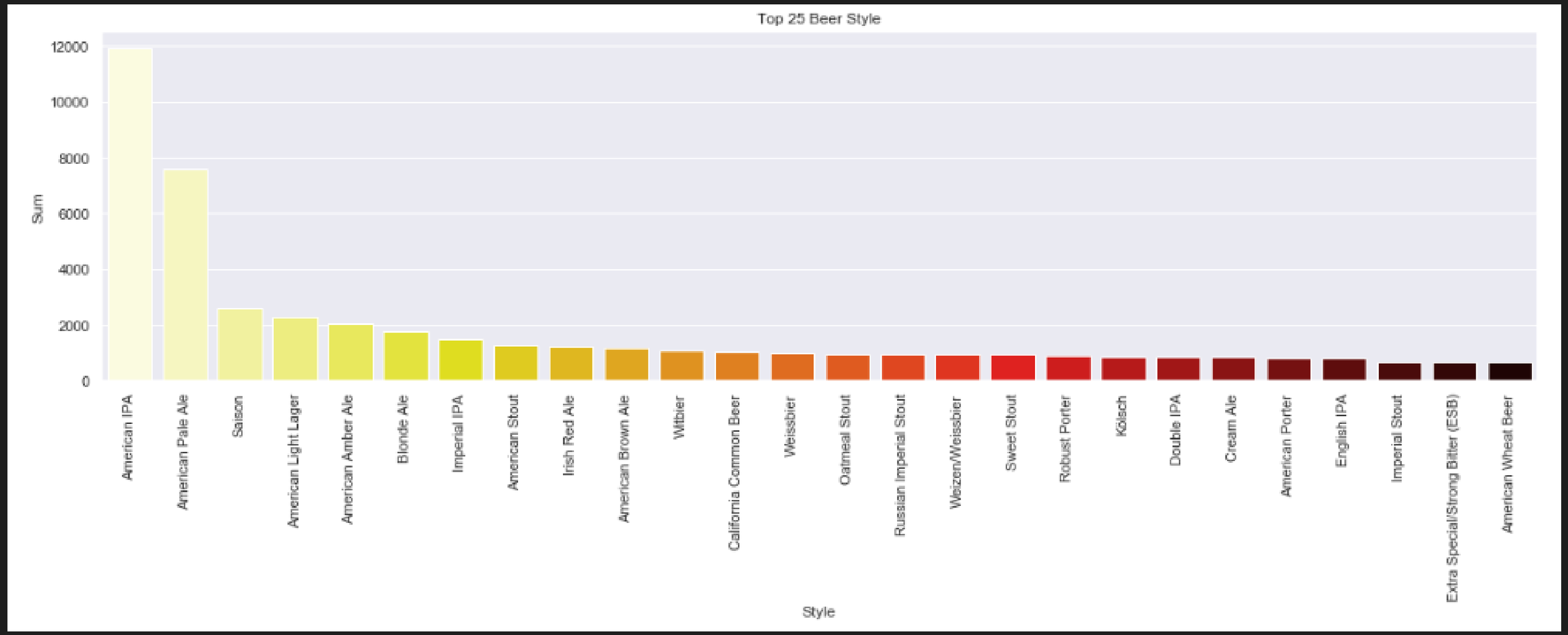
BPOM

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

```
2     55165
5      9372
3      5503
6      2963
1       250
4        11
Name: BPOM, dtype: int64
```

Exploratory Data Analysis

25 Beer Style Terbanyak



ABV Terbesar dan Terkecil

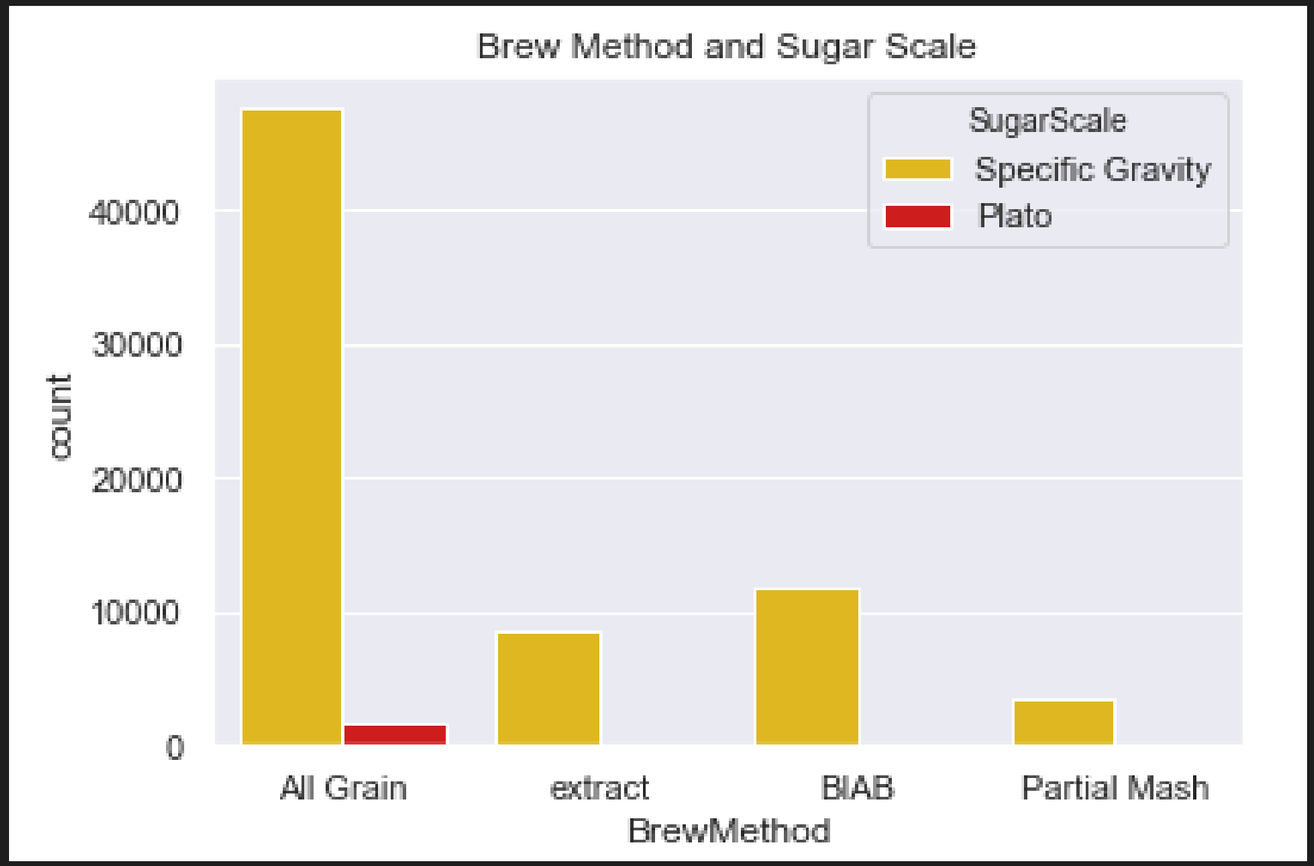
	Name	Style	StyleID	Size(L)	OG	FG	ABV
25625	Skåpsrens	American Pale Ale	10	18.0	1.5	1.083	54.72

Terbesar (54,72%)

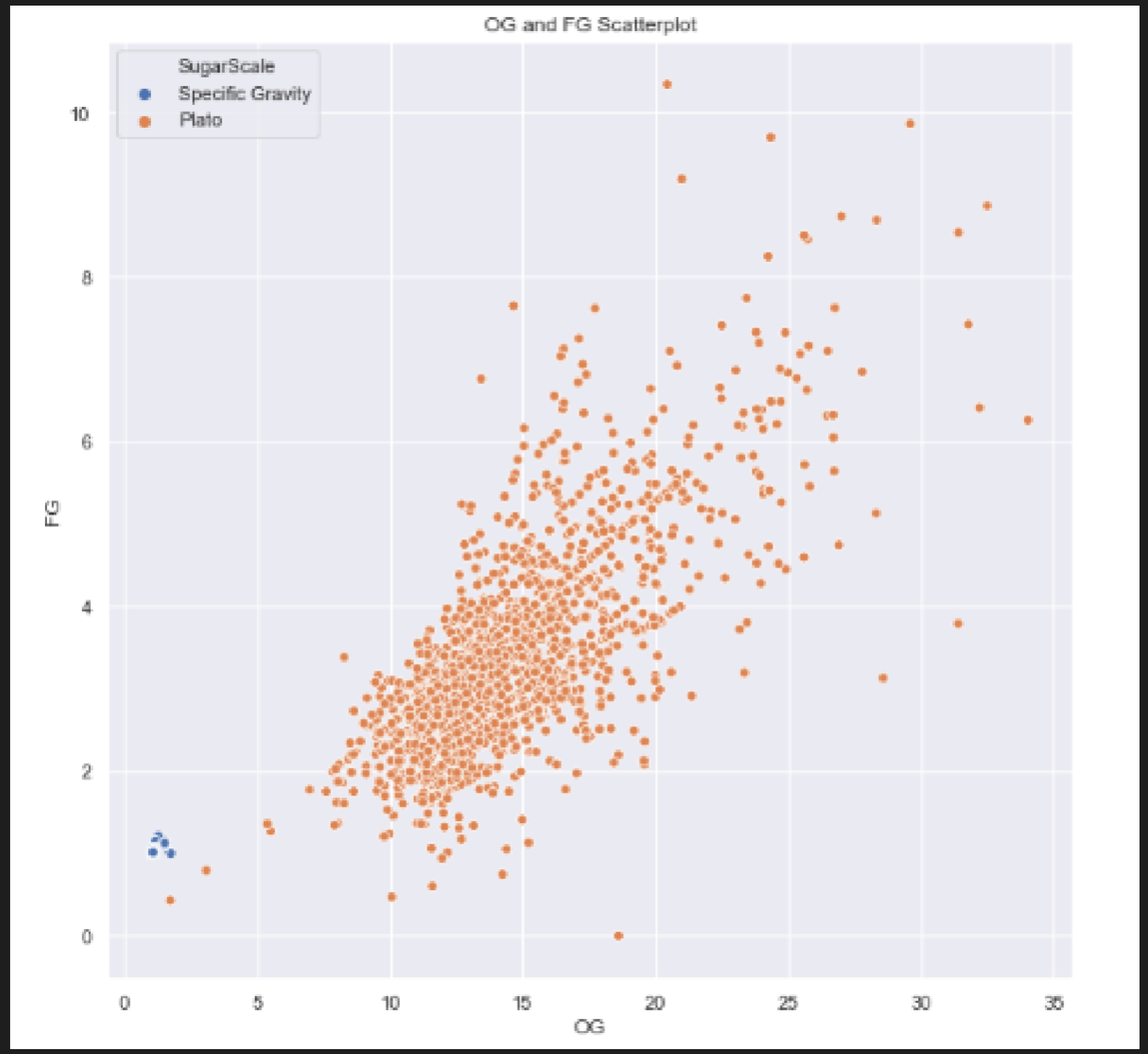
```
1238                  Reindeer Piss Holiday Ale
2229               Fergus Redmead's maple mead
3331                        Hop Rod Rye clone
4664                                    Berrr
6170     Unemployment IPA (Stone IPA Clone)
6607                                  FourPure
6732                   Mark Leyden's Recipe
7634                  Andy Krupp's Recipe 2
8559              Wheat out on the Boulevard
9626                   Zed  Krundonkulous 90
26433                     Strawberry mustard
42802                                 PH Test
54659                   Amber IPA for DELETE
69993                              JUNK RECIPE
Name: Name, dtype: object
```

Terkecil (0%)

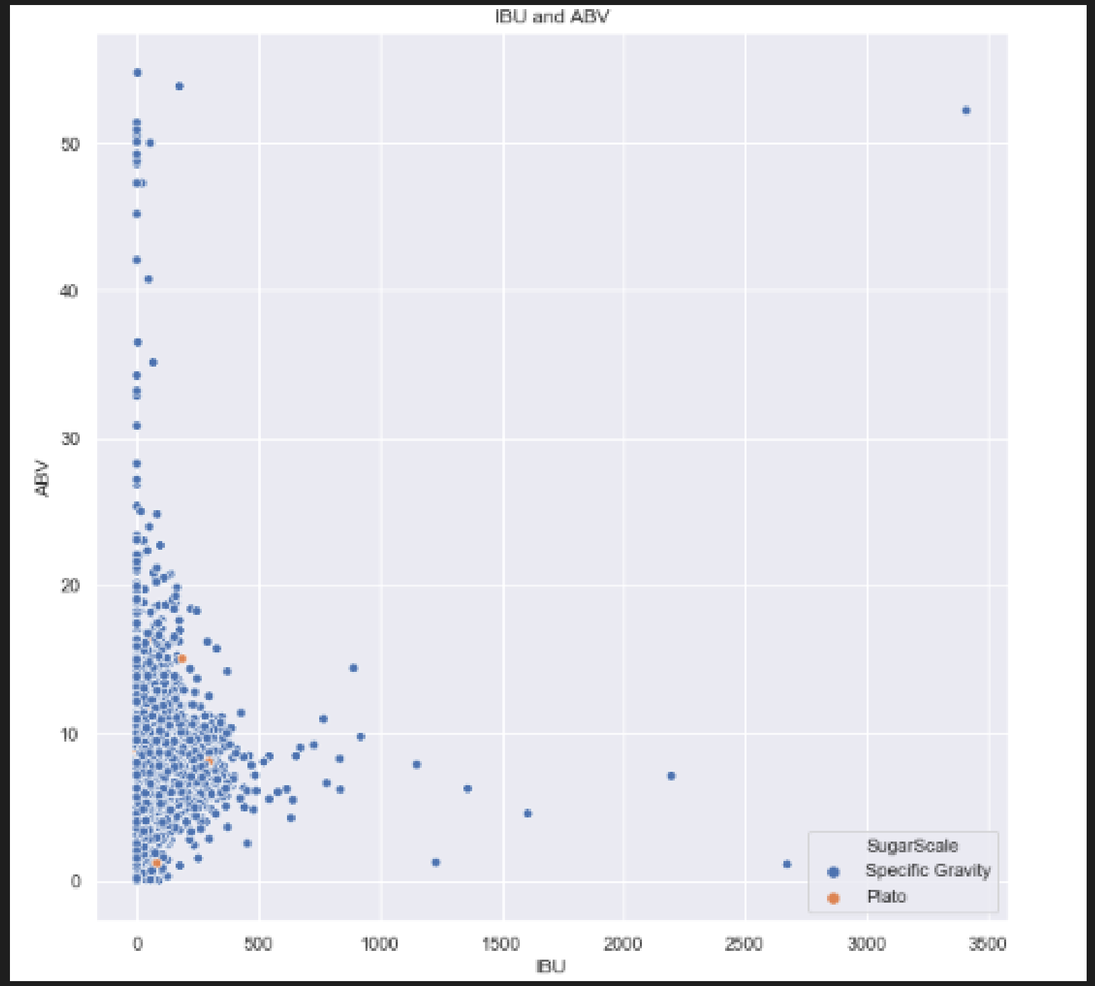
Perbandingan antar Brew Method



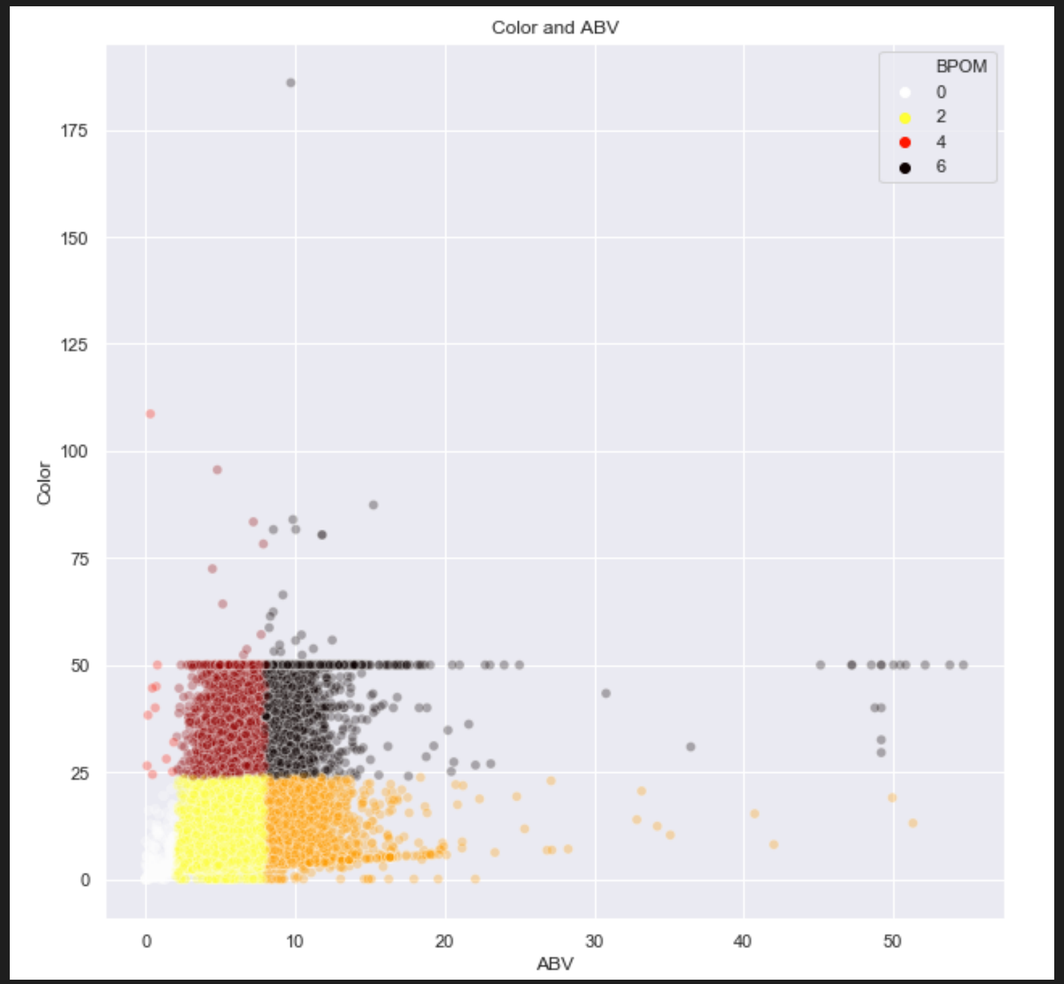
OG and FG


OG and FG Scatterplot
SugarScale
Specific Gravity
Plato
FG
OG
0
2
4
6
8
10
0
5
10
15
20
25
30
35

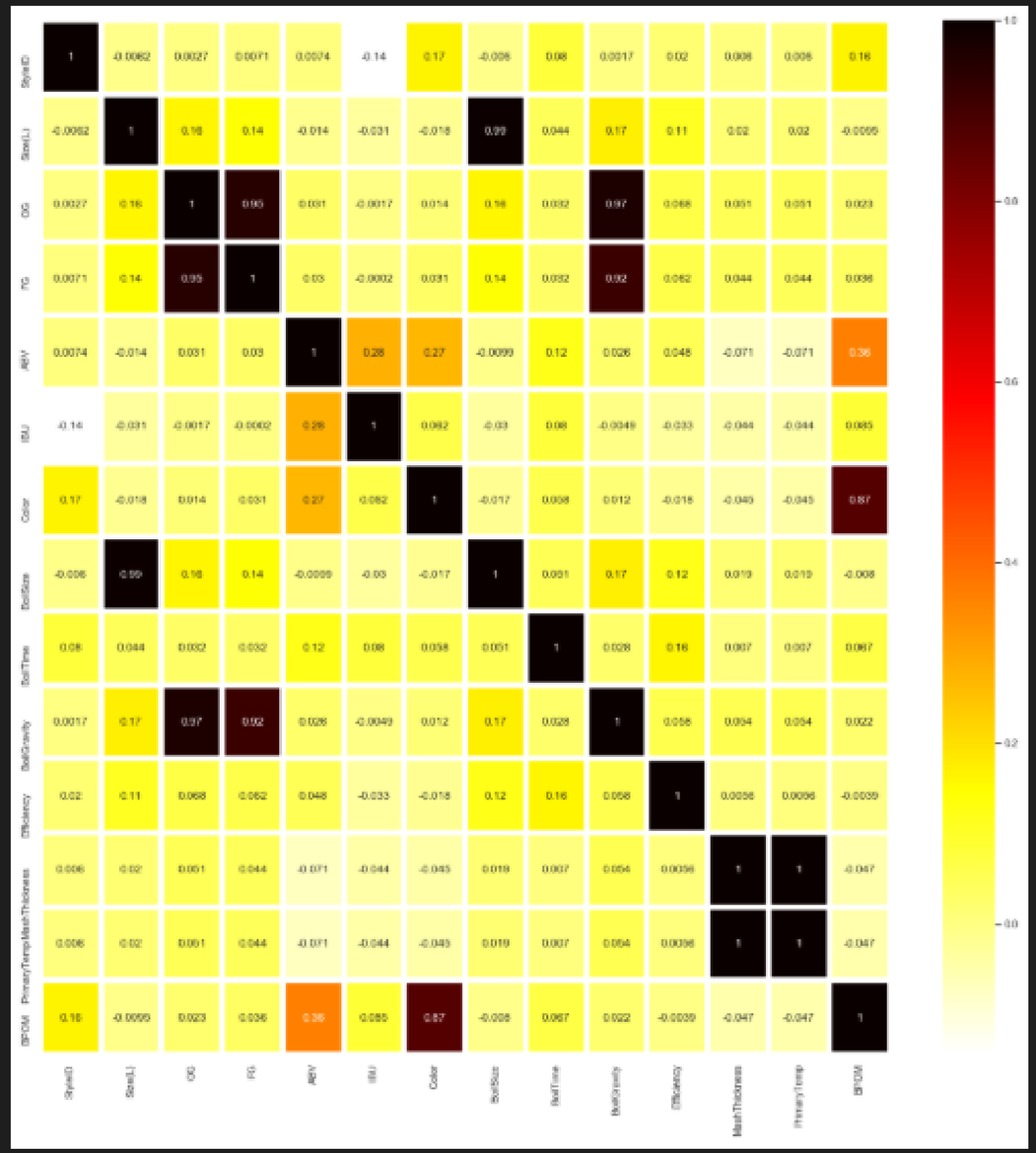
IBU and ABV


IBU and ABV
ABV
IBU
0
10
20
30
40
50
0
500
1000
1500
2000
2500
3000
3500
SugarScale
Specific Gravity
Plato

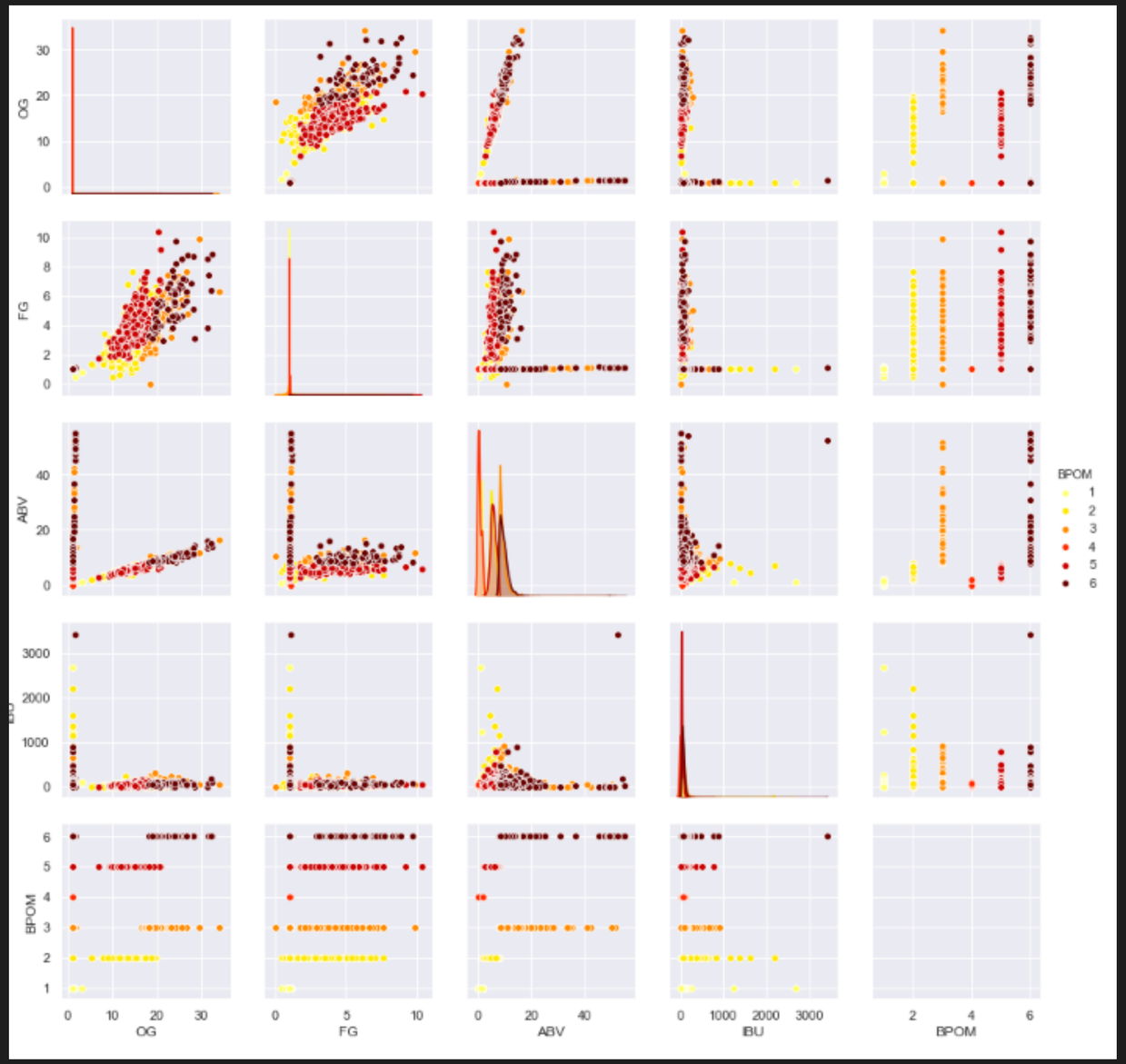
Color dan ABV



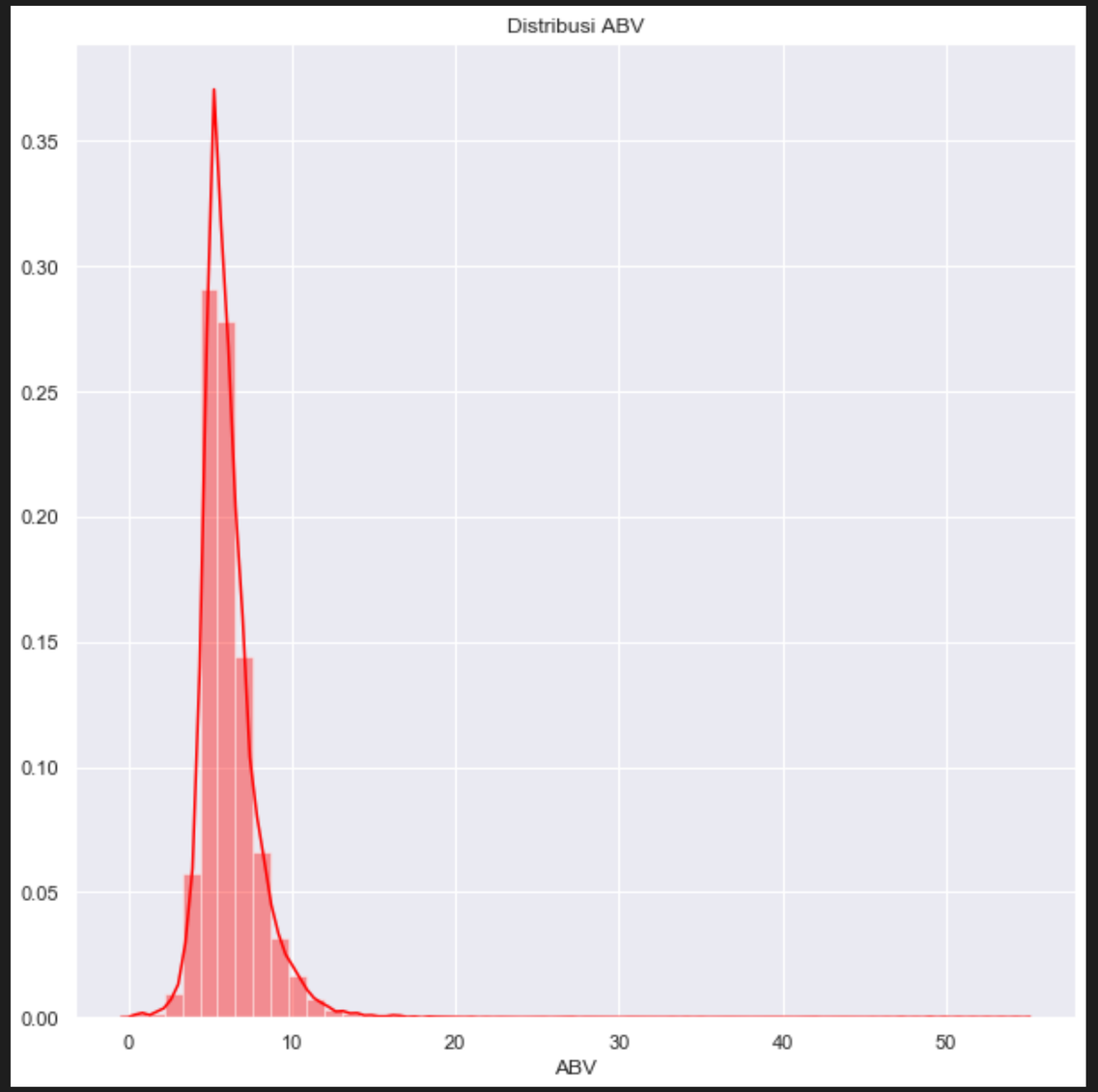
Korelasi Antar Kolom



Pairplot



Distribusi ABV



Data Analysis

1. SMOTE

Sebelum SMOTE:

```
df['BPOM'].value_counts(normalize=True)
```

```
2    0.752962
5    0.127921
3    0.075112
6    0.040443
1    0.003412
4    0.000150
Name: BPOM, dtype: float64
```

Sesudah SMOTE:

```
X_sm['BPOM'].value_counts(normalize=True)
```

```
6    0.166667
5    0.166667
4    0.166667
3    0.166667
2    0.166667
1    0.166667
Name: BPOM, dtype: float64
```

2. Train Test Split

Proses:

```
# Setting target and features
X=dfsm[['Color','ABV']]
y=dfsm['BPOM']
```

```
# Train Test Split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)
```

3. SVM

Hasil SVM:

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[16344     0     0     0     0     0]
 [10215  5791   479     0     8     0]
 [ 2209   622 13821     0     0     2]
 [    0     0     0 16709     0     0]
 [    0     0     0     2 16477   129]
 [    0     2     0     0    53 16434]]
              precision    recall  f1-score   support

           1       0.57      1.00      0.72     16344
           2       0.90      0.35      0.51     16493
           3       0.97      0.83      0.89     16654
           4       1.00      1.00      1.00     16709
           5       1.00      0.99      0.99     16608
           6       0.99      1.00      0.99     16489

    accuracy                           0.86     99297
   macro avg       0.90      0.86      0.85     99297
weighted avg       0.91      0.86      0.85     99297
```

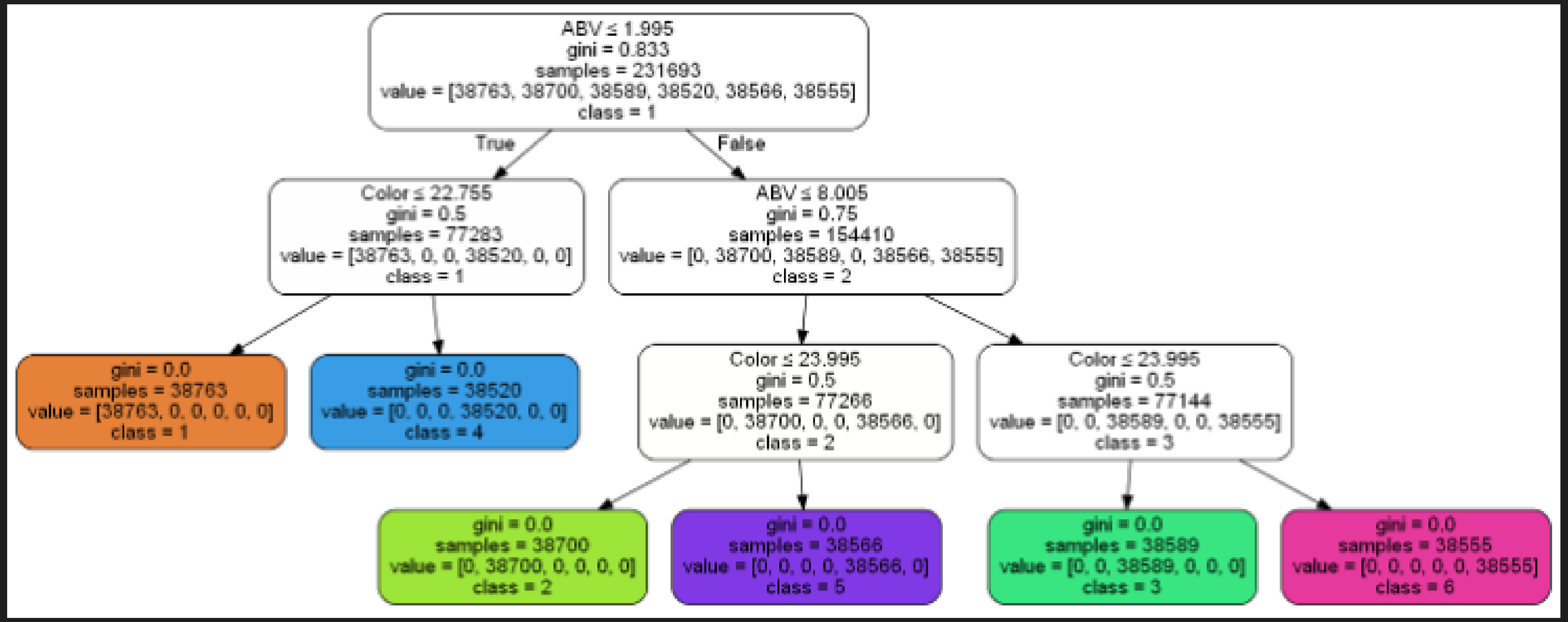
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_train, pred_train))
print(classification_report(y_train, pred_train))
```

```
[[38821     0     0     0     0     0]
 [23997 13552  1104     0    19     0]
 [ 5090  1446 31971     0     0     4]
 [    0     0     0 38456     0     0]
 [    0     0     0     9 38293   255]
 [    0     3     1     0   133 38539]]
              precision    recall  f1-score   support

           1       0.57      1.00      0.73     38821
           2       0.90      0.35      0.50     38672
           3       0.97      0.83      0.89     38511
           4       1.00      1.00      1.00     38456
           5       1.00      0.99      0.99     38557
           6       0.99      1.00      0.99     38676

    accuracy                           0.86    231693
   macro avg       0.91      0.86      0.85    231693
weighted avg       0.90      0.86      0.85    231693
```

Weight	Feature
0.4837 ± 0.0021	ABV
0.4540 ± 0.0017	Color

4. Decision Tree

Hasil Decision Tree



```
print(confusion_matrix(y_test, dtree.predict(X_test)))
print(classification_report(y_test, dtree.predict(X_test)))
```

```
[[16402     0     0     0     0     0]
 [    0 16465     0     0     0     0]
 [    0     0 16576     0     0     0]
 [    0     0     0 16645     0     0]
 [    0     0     0     0 16599     0]
 [    0     0     0     0     0 16610]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00     16402
           2       1.00      1.00      1.00     16465
           3       1.00      1.00      1.00     16576
           4       1.00      1.00      1.00     16645
           5       1.00      1.00      1.00     16599
           6       1.00      1.00      1.00     16610

    accuracy                           1.00     99297
   macro avg       1.00      1.00      1.00     99297
weighted avg       1.00      1.00      1.00     99297
```

```
print(confusion_matrix(y_train, dtree.predict(X_train)))
print(classification_report(y_train, dtree.predict(X_train)))
```

```
[[38763     0     0     0     0     0]
 [    0 38700     0     0     0     0]
 [    0     0 38589     0     0     0]
 [    0     0     0 38520     0     0]
 [    0     0     0     0 38566     0]
 [    0     0     0     0     0 38555]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00     38763
           2       1.00      1.00      1.00     38700
           3       1.00      1.00      1.00     38589
           4       1.00      1.00      1.00     38520
           5       1.00      1.00      1.00     38566
           6       1.00      1.00      1.00     38555

    accuracy                           1.00    231693
   macro avg       1.00      1.00      1.00    231693
weighted avg       1.00      1.00      1.00    231693
```

Weight	Feature
0.6675 ± 0.0026	ABV
0.5015 ± 0.0020	Color

5. Random Forest Classifier

Hasil Random Forest Classifier

```
print(confusion_matrix(y_test, rfc.predict(X_test)))
print(classification_report(y_test, rfc.predict(X_test)))
```

```
[[16402     0     0     0     0     0]
 [    0 16465     0     0     0     0]
 [    0     0 16576     0     0     0]
 [    0     0     0 16645     0     0]
 [    0     0     0     0 16599     0]
 [    0     0     0     0     0 16610]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00     16402
           2       1.00      1.00      1.00     16465
           3       1.00      1.00      1.00     16576
           4       1.00      1.00      1.00     16645
           5       1.00      1.00      1.00     16599
           6       1.00      1.00      1.00     16610

    accuracy                           1.00     99297
   macro avg       1.00      1.00      1.00     99297
weighted avg       1.00      1.00      1.00     99297
```

```
print(confusion_matrix(y_train, rfc.predict(X_train)))
print(classification_report(y_train, rfc.predict(X_train)))
```

```
[[38763     0     0     0     0     0]
 [    0 38700     0     0     0     0]
 [    0     0 38589     0     0     0]
 [    0     0     0 38520     0     0]
 [    0     0     0     0 38566     0]
 [    0     0     0     0     0 38555]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00     38763
           2       1.00      1.00      1.00     38700
           3       1.00      1.00      1.00     38589
           4       1.00      1.00      1.00     38520
           5       1.00      1.00      1.00     38566
           6       1.00      1.00      1.00     38555

    accuracy                           1.00    231693
   macro avg       1.00      1.00      1.00    231693
weighted avg       1.00      1.00      1.00    231693
```

Weight	Feature
0.6875 ± 0.0026	ABV
0.5016 ± 0.0022	Color

5. XGBOOST

Hasil XGBOOST

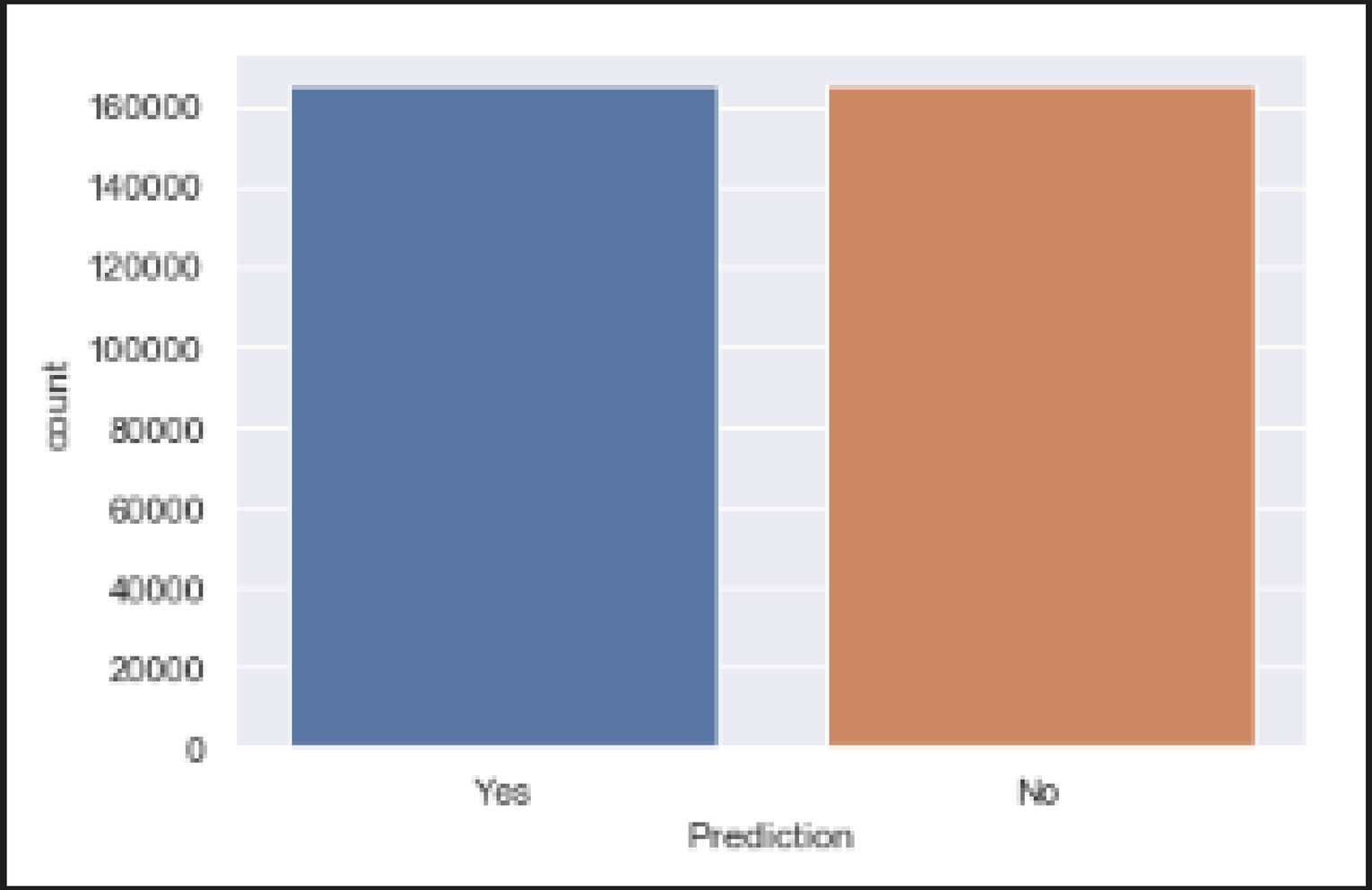
```
print(confusion_matrix(y_test, xgboost.predict(X_test)))
print(classification_report(y_test, xgboost.predict(X_test)))
```

```
[[16402     0     0     0     0     0]
 [    0 16465     0     0     0     0]
 [    0     0 16576     0     0     0]
 [    2     0     0 16643     0     0]
 [    0     0     0     0 16599     0]
 [    0     0     0     0     0 16610]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00     16402
           2       1.00      1.00      1.00     16465
           3       1.00      1.00      1.00     16576
           4       1.00      1.00      1.00     16645
           5       1.00      1.00      1.00     16599
           6       1.00      1.00      1.00     16610

    accuracy                           1.00     99297
   macro avg       1.00      1.00      1.00     99297
weighted avg       1.00      1.00      1.00     99297
```

```
print(confusion_matrix(y_train, xgboost.predict(X_train)))
print(classification_report(y_train, xgboost.predict(X_train)))
```

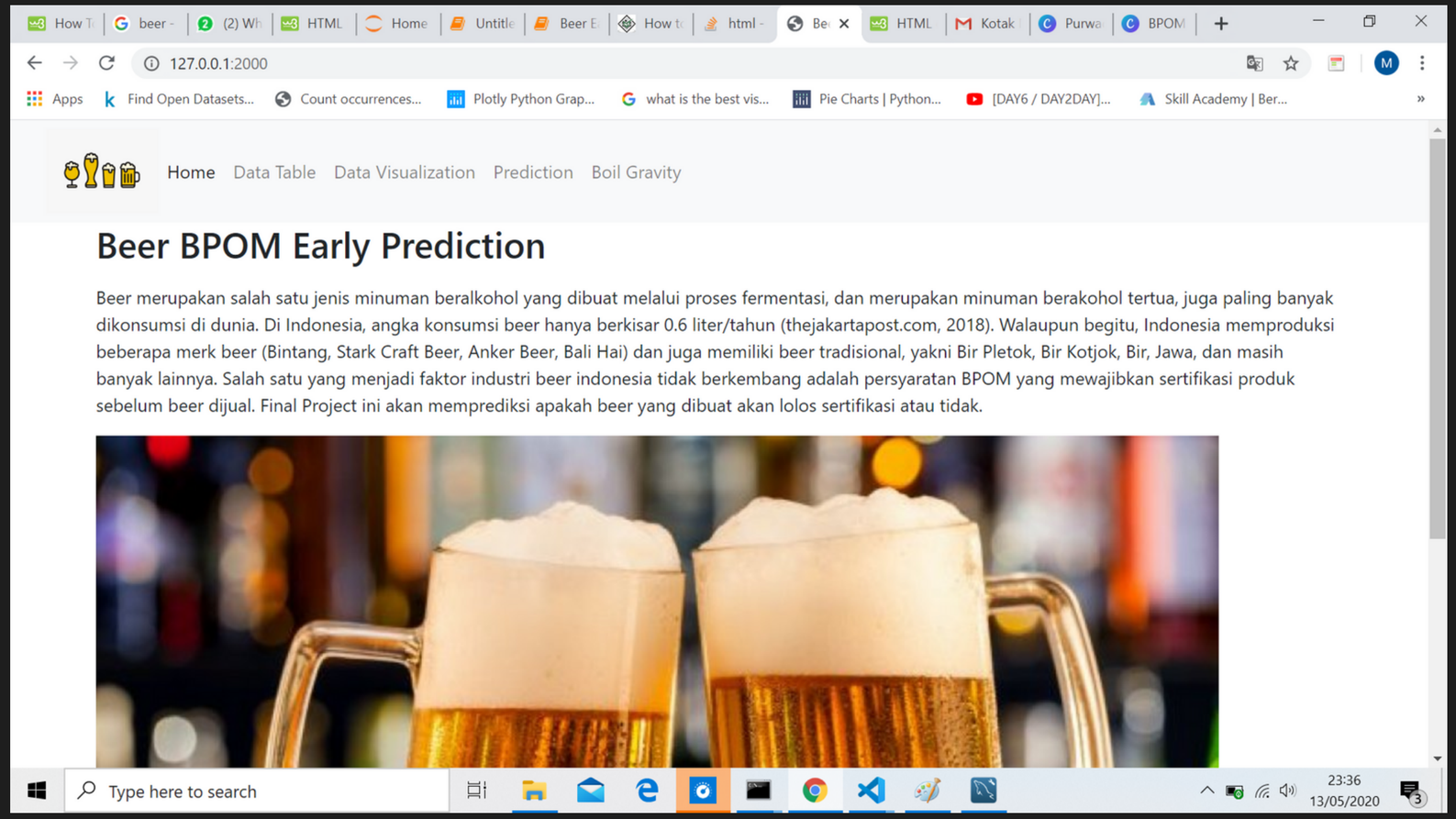
```
[[38763     0     0     0     0     0]
 [    0 38700     0     0     0     0]
 [    0     0 38589     0     0     0]
 [    0     0     0 38520     0     0]
 [    0     0     0     0 38566     0]
 [    0     0     0     0     0 38555]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00     38763
           2       1.00      1.00      1.00     38700
           3       1.00      1.00      1.00     38589
           4       1.00      1.00      1.00     38520
           5       1.00      1.00      1.00     38566
           6       1.00      1.00      1.00     38555

    accuracy                           1.00    231693
   macro avg       1.00      1.00      1.00    231693
weighted avg       1.00      1.00      1.00    231693
```

Weight	Feature
0.6684 ± 0.0026	ABV
0.4986 ± 0.0020	Color

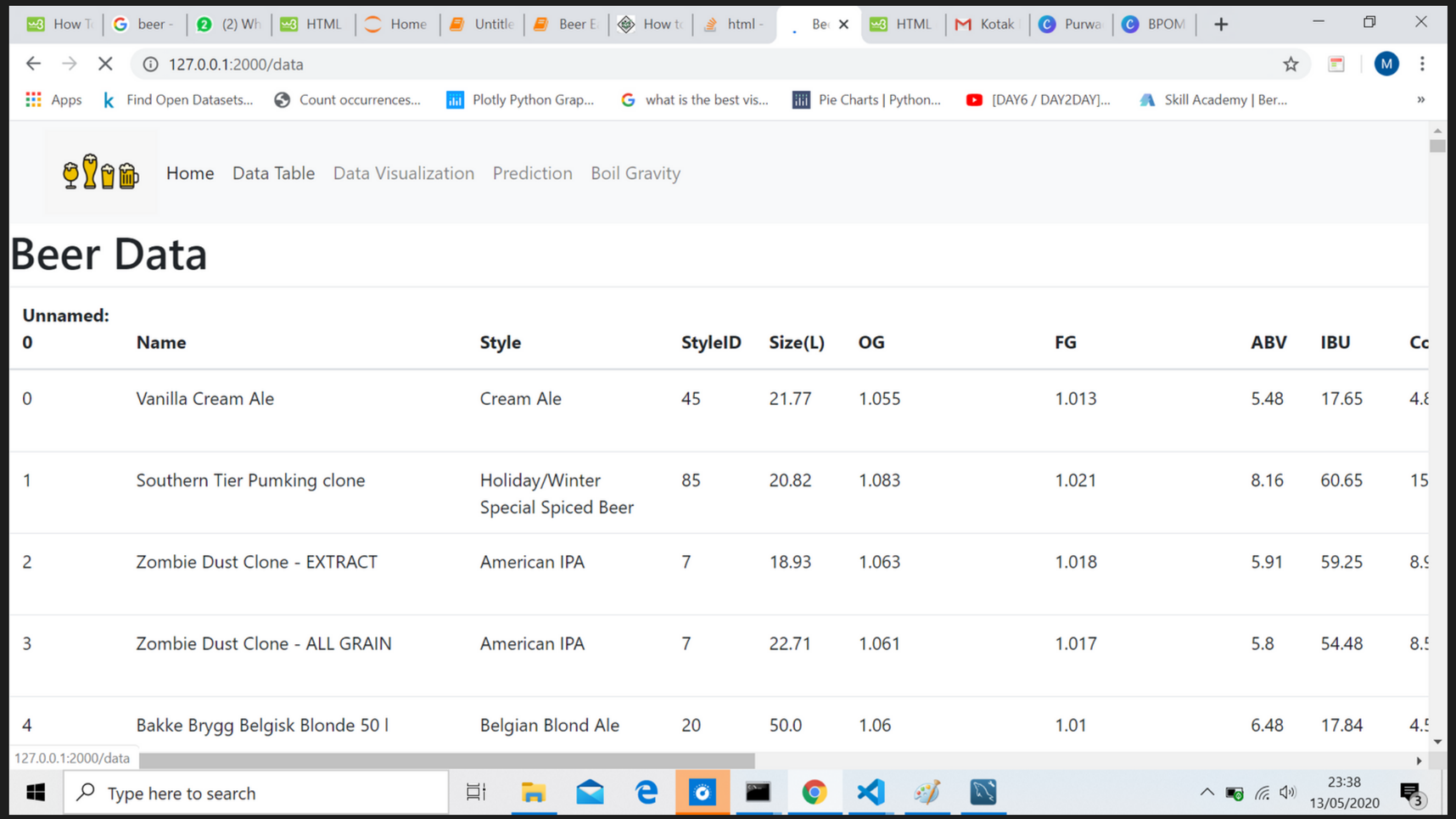

160000
140000
120000
100000
80000
60000
40000
20000
0
count
Yes
No
Prediction

Flask

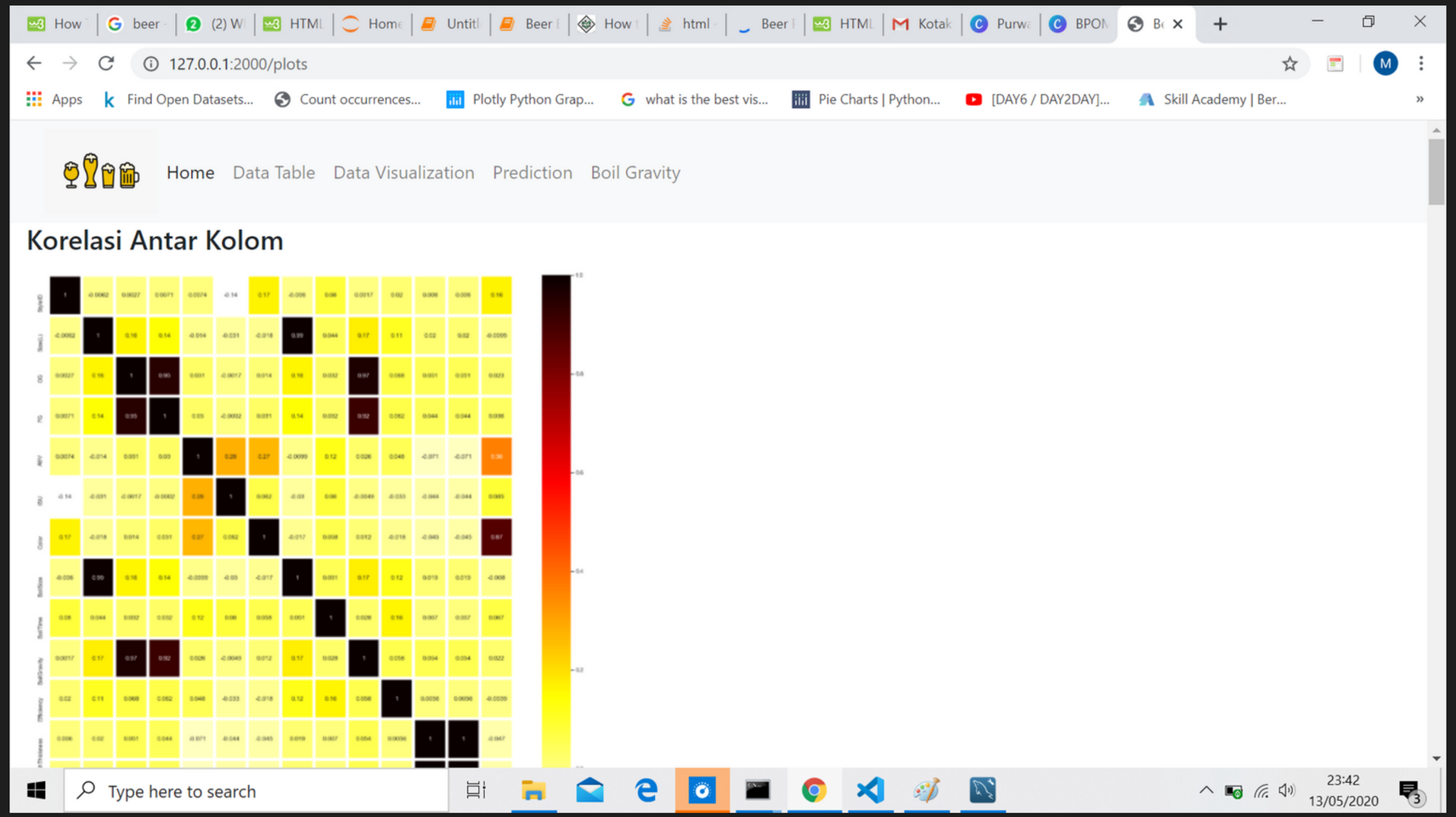
Home


127.0.0.1:2000
Home
Data Table
Data Visualization
Prediction
Boil Gravity
Beer BPOM Early Prediction
Beer merupakan salah satu jenis minuman beralkohol yang dibuat melalui proses fermentasi, dan merupakan minuman berakohol tertua, juga paling banyak dikonsumsi di dunia. Di Indonesia, angka konsumsi beer hanya berkisar 0.6 liter/tahun (thejakartapost.com, 2018). Walaupun begitu, Indonesia memproduksi beberapa merk beer (Bintang, Stark Craft Beer, Anker Beer, Bali Hai) dan juga memiliki beer tradisional, yakni Bir Pletok, Bir Kotjok, Bir, Jawa, dan masih banyak lainnya. Salah satu yang menjadi faktor industri beer indonesia tidak berkembang adalah persyaratan BPOM yang mewajibkan sertifikasi produk sebelum beer dijual. Final Project ini akan memprediksi apakah beer yang dibuat akan lolos sertifikasi atau tidak.
Type here to search
23:36
13/05/2020

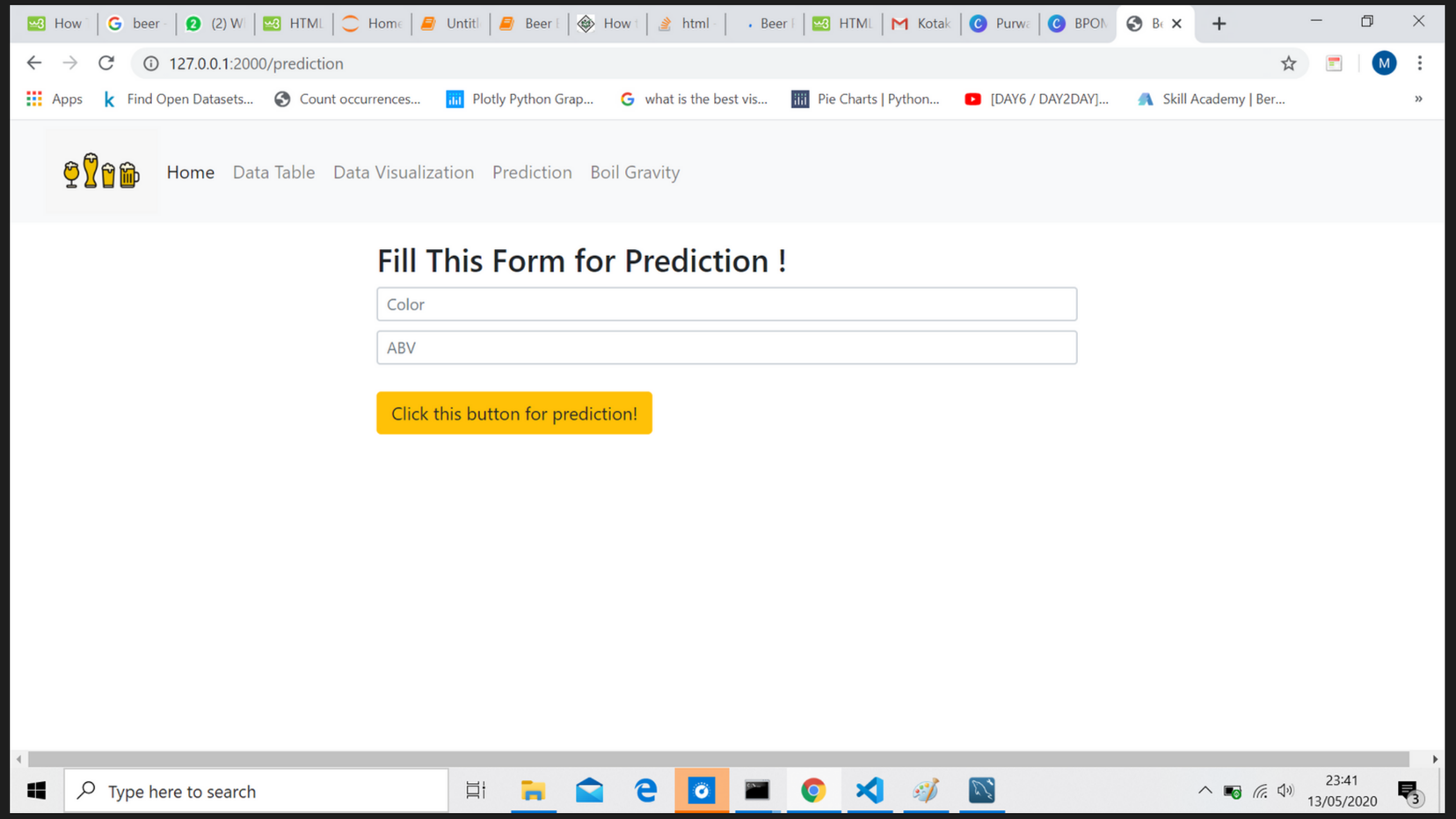
Data Table


127.0.0.1:2000/data
Home
Data Table
Data Visualization
Prediction
Boil Gravity
Beer Data
Unnamed: 0
Name
Style
StyleID
Size(L)
OG
FG
ABV
IBU
0 Vanilla Cream Ale Cream Ale 45 21.77 1.055 1.013 5.48 17.65
1 Southern Tier Pumking clone Holiday/Winter Special Spiced Beer 85 20.82 1.083 1.021 8.16 60.65
2 Zombie Dust Clone - EXTRACT American IPA 7 18.93 1.063 1.018 5.91 59.25
3 Zombie Dust Clone - ALL GRAIN American IPA 7 22.71 1.061 1.017 5.8 54.48
4 Bakke Brygg Belgisk Blonde 50 l Belgian Blond Ale 20 50.0 1.06 1.01 6.48 17.84
Type here to search
23:38
13/05/2020

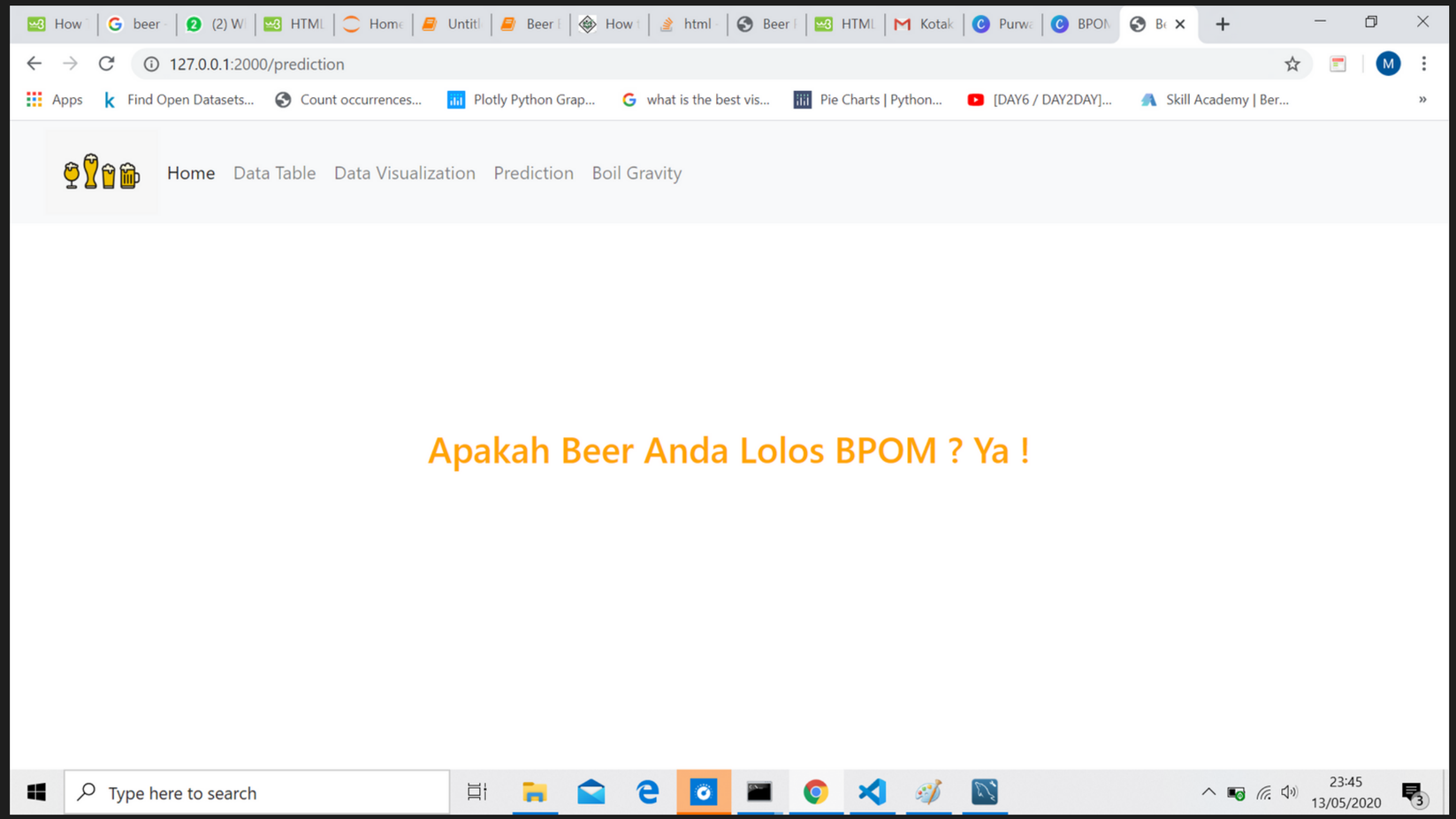
Data Visualisasi



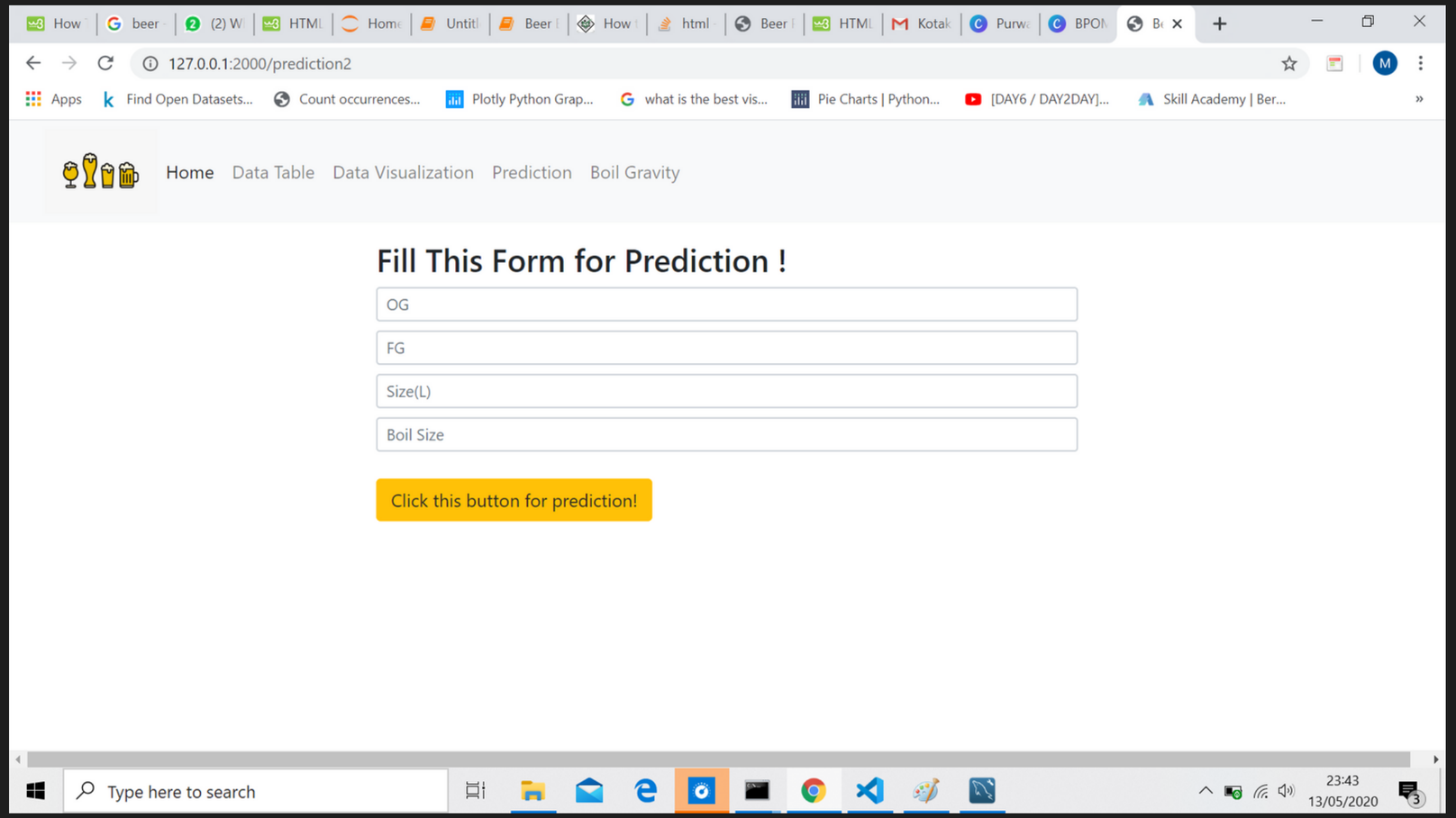
BPOM Prediction



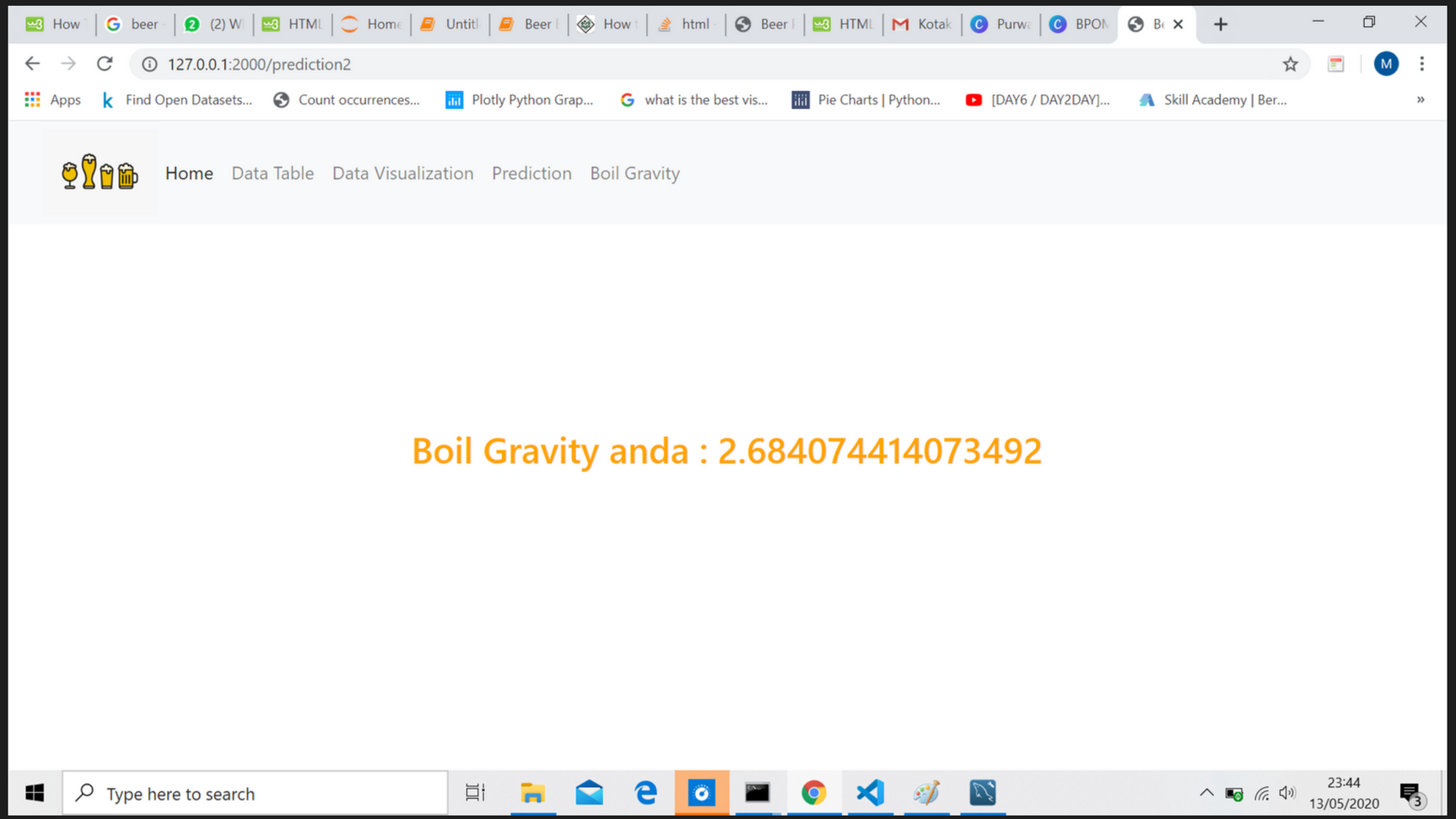
Result


127.0.0.1:2000/prediction
Home
Data Table
Data Visualization
Prediction
Boil Gravity
Apakah Beer Anda Lolos BPOM ? Ya !
Type here to search
23:45
13/05/2020

Boil Gravity Prediction



Result


127.0.0.1:2000/prediction2
Apps
Find Open Datasets...
Count occurrences...
Plotly Python Grap...
what is the best vis...
Pie Charts | Python...
[DAY6 / DAY2DAY]...
Skill Academy | Ber...
Home
Data Table
Data Visualization
Prediction
Boil Gravity
Boil Gravity anda : 2.684074414073492
Type here to search
23:44
13/05/2020

Kesimpulan

Terimakasih