Importing The Libraries

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Exploratory Data Analysis

In [32]:
```
data = pd.read_csv(r"C:\Users\NARESH SANA\Downloads\cosmetics.csv")
data
```

Out[32]:

	Label	Brand	Name	Price	Rank	Ingredients	Combination	Dry	Norma
0	Moisturizer	LA MER	Crème de la Mer	175	4.1	Algae (Seaweed) Extract, Mineral Oil, Petrolat...	1	1	1
1	Moisturizer	SK-II	Facial Treatment Essence	179	4.1	Galactomyces Ferment Filtrate (Pitera), Butyle...	1	1	1
2	Moisturizer	DRUNK ELEPHANT	Protini™ Polypeptide Cream	68	4.4	Water, Dicaprylyl Carbonate, Glycerin, Ceteary...	1	1	1
3	Moisturizer	LA MER	The Moisturizing Soft Cream	175	3.8	Algae (Seaweed) Extract, Cyclopentasiloxane, P...	1	1	1
4	Moisturizer	IT COSMETICS	Your Skin But Better™ CC+™ Cream with SPF 50+	38	4.1	Water, Snail Secretion Filtrate, Phenyl Trimet...	1	1	1
...	...	...	...	...	...	...	...	...	..
1467	Sun protect	KORRES	Yoghurt Nourishing Fluid Veil Face Sunscreen B...	35	3.9	Water, Alcohol Denat., Potassium Cetyl Phospha...	1	1	1
1468	Sun protect	KATE SOMERVILLE	Daily Deflector™ Waterlight Broad Spectrum SPF...	48	3.6	Water, Isododecane, Dimethicone, Butyloctyl Sa...	0	0	(
1469	Sun protect	VITA LIBERATA	Self Tan Dry Oil SPF 50	54	3.5	Water, Dihydroxyacetone, Glycerin, Sclerocarya...	0	0	(
1470	Sun protect	ST. TROPEZ TANNING ESSENTIALS	Pro Light Self Tan Bronzing Mist	20	1.0	Water, Dihydroxyacetone, Propylene Glycol, PPG...	0	0	(
1471	Sun protect	DERMAFLASH	DERMAPROTECT Daily Defense Broad Spectrum SPF 50+	45	0.0	Visit the DERMAFLASH boutique	1	1	1

1472 rows × 11 columns

In [34]:
```
data.shape
```

Out[34]:
```
(1472, 11)
```

In [36]:
```
data.describe()
```

Out[36]:

	Price	Rank	Combination	Dry	Normal	Oily	Sensitive
count	1472.000000	1472.000000	1472.00000	1472.000000	1472.000000	1472.000000	1472.000000
mean	55.584239	4.153261	0.65625	0.614130	0.652174	0.607337	0.513587
std	45.014429	0.633918	0.47512	0.486965	0.476442	0.488509	0.499985
min	3.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000
25%	30.000000	4.000000	0.00000	0.000000	0.000000	0.000000	0.000000
50%	42.500000	4.300000	1.00000	1.000000	1.000000	1.000000	1.000000
75%	68.000000	4.500000	1.00000	1.000000	1.000000	1.000000	1.000000
max	370.000000	5.000000	1.00000	1.000000	1.000000	1.000000	1.000000

In [38]:
```
data.head(5)
```

Out[38]:

	Label	Brand	Name	Price	Rank	Ingredients	Combination	Dry	Normal	Oily	Ser
0	Moisturizer	LA MER	Crème de la Mer	175	4.1	Algae (Seaweed) Extract, Mineral Oil, Petrolat...	1	1	1	1	
1	Moisturizer	SK-II	Facial Treatment Essence	179	4.1	Galactomyces Ferment Filtrate (Pitera), Butyle...	1	1	1	1	
2	Moisturizer	DRUNK ELEPHANT	Protini™ Polypeptide Cream	68	4.4	Water, Dicaprylyl Carbonate, Glycerin, Ceteary...	1	1	1	1	
3	Moisturizer	LA MER	The Moisturizing Soft Cream	175	3.8	Algae (Seaweed) Extract, Cyclopentasiloxane, P...	1	1	1	1	
4	Moisturizer	IT COSMETICS	Your Skin But Better™ CC+™ Cream with SPF 50+	38	4.1	Water, Snail Secretion Filtrate, Phenyl Trimet...	1	1	1	1	

In [40]:
```
data.tail(5)
```

Out[40]:

	Label	Brand	Name	Price	Rank	Ingredients	Combination	Dry	Normal	Oil
1467	Sun protect	KORRES	Yoghurt Nourishing Fluid Veil Face Sunscreen B...	35	3.9	Water, Alcohol Denat., Potassium Cetyl Phospha...	1	1	1	
1468	Sun protect	KATE SOMERVILLE	Daily Deflector™ Waterlight Broad Spectrum SPF...	48	3.6	Water, Isododecane, Dimethicone, Butyloctyl Sa...	0	0	0	
1469	Sun protect	VITA LIBERATA	Self Tan Dry Oil SPF 50	54	3.5	Water, Dihydroxyacetone, Glycerin, Sclerocarya...	0	0	0	
1470	Sun protect	ST. TROPEZ TANNING ESSENTIALS	Pro Light Self Tan Bronzing Mist	20	1.0	Water, Dihydroxyacetone, Propylene Glycol, PPG...	0	0	0	
1471	Sun protect	DERMAFLASH	DERMAPROTECT Daily Defense Broad Spectrum SPF 50+	45	0.0	Visit the DERMAFLASH boutique	1	1	1	

In [42]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1472 entries, 0 to 1471
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Label        1472 non-null   object 
 1   Brand        1472 non-null   object 
 2   Name         1472 non-null   object 
 3   Price        1472 non-null   int64  
 4   Rank         1472 non-null   float64
 5   Ingredients  1472 non-null   object 
 6   Combination  1472 non-null   int64  
 7   Dry          1472 non-null   int64  
 8   Normal       1472 non-null   int64  
 9   Oily         1472 non-null   int64  
 10  Sensitive    1472 non-null   int64  
dtypes: float64(1), int64(6), object(4)
memory usage: 126.6+ KB
```

In [44]:
```
data.isnull().sum()
```

Out[44]:
```
Label          0
Brand          0
Name           0
Price          0
Rank           0
Ingredients    0
Combination    0
Dry            0
Normal         0
Oily           0
Sensitive      0
dtype: int64
```

In [46]:
```
data.min()
```

Out[46]:
```
Label                                          Cleanser
Brand                                          ALGENIST
Name           #GLITTERMASK GRAVITYMUD™ Firming Treatment
Price                                                 3
Rank                                                0.0
Ingredients                                      #NAME?
Combination                                           0
Dry                                                   0
Normal                                                0
Oily                                                  0
Sensitive                                             0
dtype: object
```

In [48]:
```
data.max()
```

Out[48]:
```
Label                                                 Treatment
Brand                                        YVES SAINT LAURENT
Name           Énergie de Vie The Smoothing & Plumping Water-...
Price                                                       370
Rank                                                        5.0
Ingredients    Zingiber Officinale (Ginger) Water, Water, Gly...
Combination                                                   1
Dry                                                           1
Normal                                                        1
Oily                                                          1
Sensitive                                                     1
dtype: object
```

In [50]:
```
list(data)
```

Out[50]:
```
['Label',
 'Brand',
 'Name',
 'Price',
 'Rank',
 'Ingredients',
 'Combination',
 'Dry',
 'Normal',
 'Oily',
 'Sensitive']
```

Dropping unwanted columns

In [52]:
```
data1 = data.drop(['Name', 'Ingredients'],axis=1)
data1
```

Out[52]:

	Label	Brand	Price	Rank	Combination	Dry	Normal	Oily	Sensitive
0	Moisturizer	LA MER	175	4.1	1	1	1	1	1
1	Moisturizer	SK-II	179	4.1	1	1	1	1	1
2	Moisturizer	DRUNK ELEPHANT	68	4.4	1	1	1	1	0
3	Moisturizer	LA MER	175	3.8	1	1	1	1	1
4	Moisturizer	IT COSMETICS	38	4.1	1	1	1	1	1
...	...	...	...	...	...	...	...	...	...
1467	Sun protect	KORRES	35	3.9	1	1	1	1	1
1468	Sun protect	KATE SOMERVILLE	48	3.6	0	0	0	0	0
1469	Sun protect	VITA LIBERATA	54	3.5	0	0	0	0	0
1470	Sun protect	ST. TROPEZ TANNING ESSENTIALS	20	1.0	0	0	0	0	0
1471	Sun protect	DERMAFLASH	45	0.0	1	1	1	1	1

1472 rows × 9 columns

In [54]:
```
data1.isnull().sum()
```

Out[54]:
```
Label          0
Brand          0
Price          0
Rank           0
Combination    0
Dry            0
Normal         0
Oily           0
Sensitive      0
dtype: int64
```

Getting Dummies

In [56]:
```
data1 = pd.get_dummies(data1, dtype=int)
data1
```

Out[56]:

	Price	Rank	Combination	Dry	Normal	Oily	Sensitive	Label_Cleanser	Label_Eye cream	Label_Face Mask	...	Br
0	175	4.1	1	1	1	1	1	0	0	0	...	
1	179	4.1	1	1	1	1	1	0	0	0	...	
2	68	4.4	1	1	1	1	0	0	0	0	...	
3	175	3.8	1	1	1	1	1	0	0	0	...	
4	38	4.1	1	1	1	1	1	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1467	35	3.9	1	1	1	1	1	0	0	0	...	
1468	48	3.6	0	0	0	0	0	0	0	0	...	
1469	54	3.5	0	0	0	0	0	0	0	0	...	
1470	20	1.0	0	0	0	0	0	0	0	0	...	
1471	45	0.0	1	1	1	1	1	0	0	0	...	

1472 rows × 129 columns

In [58]:
```
data.shape
```

Out[58]:
```
(1472, 11)
```

In [60]:
```
list(data1)
```

```
 'Brand_SKIN LAUNDRY',
 'Brand_SMASHBOX',
 'Brand_SON & PARK',
 'Brand_ST. TROPEZ TANNING ESSENTIALS',
 'Brand_SUMMER FRIDAYS',
 'Brand_SUNDAY RILEY',
 'Brand_SUPERGOOP!',
 'Brand_TARTE',
 'Brand_TATA HARPER',
 'Brand_TATCHA',
 'Brand_TOM FORD',
 'Brand_TOO COOL FOR SCHOOL',
 'Brand_TOO FACED',
 'Brand_URBAN DECAY',
 'Brand_VITA LIBERATA',
 'Brand_VOLITION BEAUTY',
 'Brand_WANDER BEAUTY',
 'Brand_YOUTH TO THE PEOPLE',
 'Brand_YVES SAINT LAURENT']
```

In [62]:
```
data1.isnull().sum()
```

Out[62]:
```
Price                        0
Rank                         0
Combination                  0
Dry                          0
Normal                       0
                            ..
Brand_VITA LIBERATA          0
Brand_VOLITION BEAUTY        0
Brand_WANDER BEAUTY          0
Brand_YOUTH TO THE PEOPLE    0
Brand_YVES SAINT LAURENT     0
Length: 129, dtype: int64
```

Correlation

In [64]:
```
cor_mat = data1.corr()
cor_mat
```

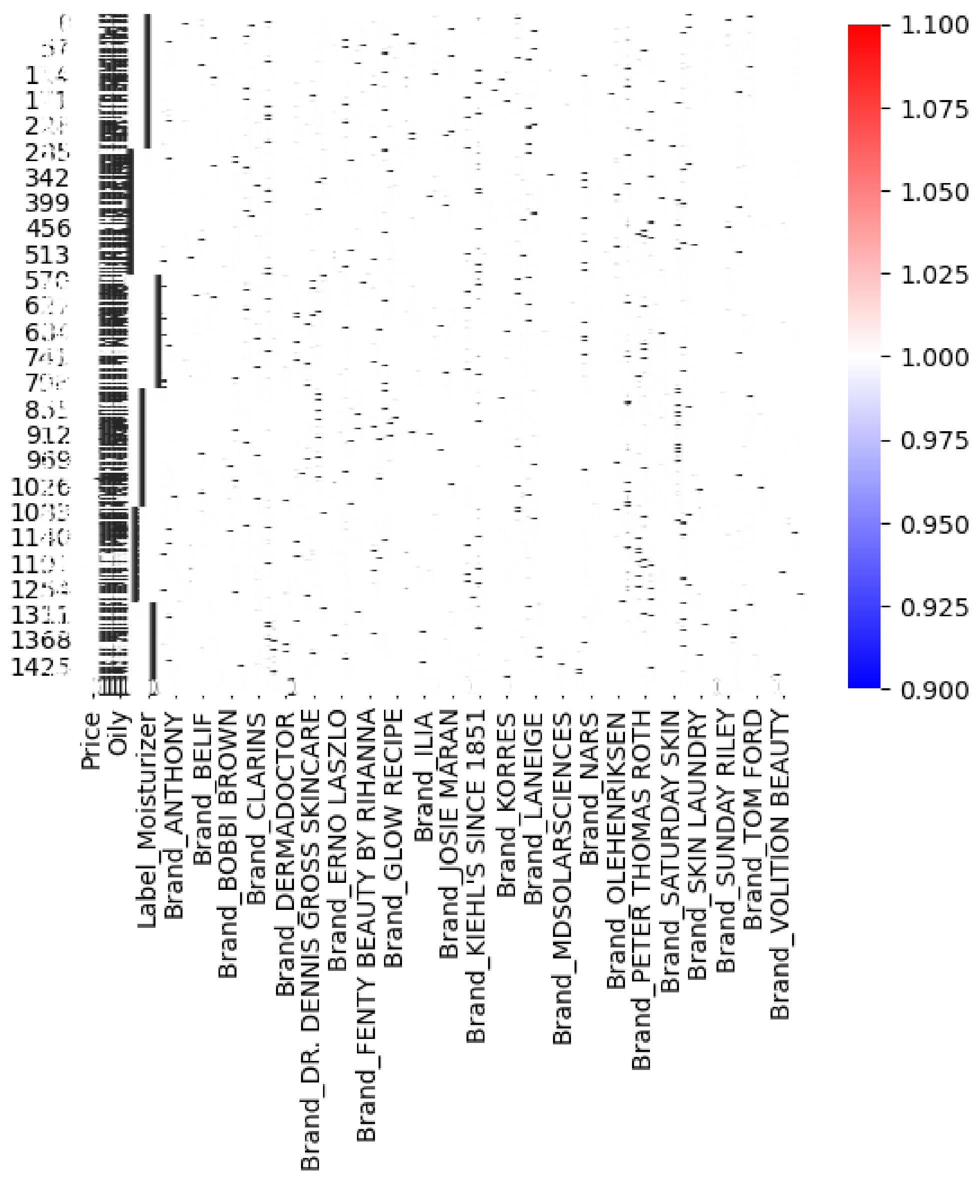
Out[64]:

	Price	Rank	Combination	Dry	Normal	Oily	Sensitive	Label_Cleanse
Price	1.000000	-0.025215	0.012575	0.065525	0.049230	0.003978	0.007621	-0.24808
Rank	-0.025215	1.000000	0.036904	0.026982	0.051926	0.021041	0.015946	0.12202
Combination	0.012575	0.036904	1.000000	0.830784	0.927966	0.882528	0.689316	-0.08518
Dry	0.065525	0.026982	0.830784	1.000000	0.874436	0.745767	0.722367	-0.11565
Normal	0.049230	0.051926	0.927966	0.874436	1.000000	0.835227	0.713320	-0.09893
...	...	...	...	...	...	...	...	.
Brand_VITA LIBERATA	-0.004987	-0.017652	-0.050965	-0.046534	-0.050508	-0.045873	-0.037902	-0.01791
Brand_VOLITION BEAUTY	-0.008023	0.024574	0.042253	0.046276	0.042635	0.046942	0.033454	0.00135
Brand_WANDER BEAUTY	-0.025070	0.017272	0.026696	0.029238	0.026937	0.029659	-0.001003	-0.01791
Brand_YOUTH TO THE PEOPLE	-0.013625	0.022239	0.029237	0.014222	0.029748	0.015138	0.047514	-0.00844
Brand_YVES SAINT LAURENT	0.010091	-0.030992	-0.036025	-0.032893	-0.035702	-0.032426	-0.026792	-0.01266

129 rows × 129 columns

```
In [66]: sns.heatmap(data1,vmax=1,vmin=1,annot=True,linewidth=5,cmap='bwr')
```

```
Out[66]: <Axes: >
```



In [109]:
```
y = data1['Price']
x = data1.drop('Price', axis=1)
data1
```

Out[109]:

	Price	Rank	Combination	Dry	Normal	Oily	Sensitive	Label_Cleanser	Label_Eye cream	Label_Face Mask	...	B
0	175	4.1	1	1	1	1	1	0	0	0	...	
1	179	4.1	1	1	1	1	1	0	0	0	...	
2	68	4.4	1	1	1	1	0	0	0	0	...	
3	175	3.8	1	1	1	1	1	0	0	0	...	
4	38	4.1	1	1	1	1	1	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1467	35	3.9	1	1	1	1	1	0	0	0	...	
1468	48	3.6	0	0	0	0	0	0	0	0	...	
1469	54	3.5	0	0	0	0	0	0	0	0	...	
1470	20	1.0	0	0	0	0	0	0	0	0	...	
1471	45	0.0	1	1	1	1	1	0	0	0	...	

1472 rows × 129 columns

In [99]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split (x, y, test_size=20, random_state=49
```

In [100]:
```
x_train
```

Out[100]:

	Rank	Combination	Dry	Normal	Oily	Sensitive	Label_Cleanser	Label_Eye cream	Label_Face Mask	Label_Moistu
232	4.7	0	0	0	0	0	0	0	0	
762	4.3	1	1	1	1	1	0	0	0	
706	3.8	1	1	1	1	1	0	0	0	
17	4.4	1	1	1	1	1	0	0	0	
540	4.9	1	1	1	1	1	1	0	0	
...	...	...	...	...	...	...	...	...	...	
453	4.5	0	0	0	0	0	1	0	0	
908	4.4	1	1	1	1	1	0	0	1	
1206	3.5	1	1	1	1	1	0	1	0	
424	4.6	0	0	0	0	0	1	0	0	
426	4.2	0	0	0	0	0	1	0	0	

1452 rows × 128 columns

In [101]:
```
x_test
```

Out[101]:

	Rank	Combination	Dry	Normal	Oily	Sensitive	Label_Cleanser	Label_Eye cream	Label_Face Mask	Label_Moistu
1104	4.4	1	1	1	1	0	0	1	0	
1263	3.0	1	1	1	1	1	0	1	0	
897	3.9	1	1	1	1	0	0	0	1	
1408	4.4	0	0	0	0	0	0	0	0	
1120	3.5	0	0	0	0	0	0	1	0	
1281	4.1	0	0	0	0	0	0	1	0	
923	4.1	1	0	1	1	0	0	0	1	
1469	3.5	0	0	0	0	0	0	0	0	
50	4.5	1	1	1	1	1	0	0	0	
282	4.2	1	1	1	1	1	0	0	0	
1392	3.4	1	0	1	1	0	0	0	0	
1446	3.3	1	1	1	1	1	0	0	0	
1378	4.8	1	1	1	1	1	0	0	0	
195	4.6	1	1	1	1	1	0	0	0	
33	4.5	0	1	0	1	1	0	0	0	
300	4.4	0	0	0	0	0	1	0	0	
1464	4.1	1	1	1	1	1	0	0	0	
1238	3.3	0	0	0	0	0	0	1	0	
1038	3.4	1	1	1	1	1	0	0	1	
120	4.3	0	0	0	0	0	0	0	0	

20 rows × 128 columns

In [102]:
```
y_train
```

Out[102]:
```
232     58
762     48
706     48
17      39
540     35
        ..
453     25
908     46
1206    65
424     22
426      7
Name: Price, Length: 1452, dtype: int64
```

In [103]:
```
y_test
```

Out[103]:
```
1104     65
1263    105
897      59
1408     50
1120    255
1281     42
923       6
1469     54
50       10
282      29
1392     36
1446     26
1378     55
195      39
33       48
300      38
1464     38
1238     65
1038      7
120     127
Name: Price, dtype: int64
```

Random Forest Regression

In [104]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
import pandas as pd
n_estimators = [50, 100, 200]
criterion = ['gini', 'entropy']
max_depth = [3, 5, 10]
parameters = {'n_estimators': n_estimators, 'criterion': criterion, 'max_depth': max_dep
RFC_cls = RandomForestClassifier()
grid_search = GridSearchCV(RFC_cls, parameters, cv=5)
grid_search.fit(x_train, y_train)
```

Out[104]:
```
▸              GridSearchCV
▸ estimator: RandomForestClassifier
        ▸ RandomForestClassifier
```

In [105]:
```
grid_search.best_params_
```

Out[105]:
```
{'criterion': 'gini', 'max_depth': 10, 'n_estimators': 50}
```

In [106]:
```
y_pred=grid_search.predict(x_test)
y_pred
```

Out[106]:
```
array([ 60,  98,  59,  55, 215,  38,  20,  45, 140,  29,  36,  28,  32,
        48,  48,  62,  38,  38,   9,  63], dtype=int64)
```

Confusion Matrix

In [107]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

Out[107]:
```
array([[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],
       [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, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0],
       [0, 0, 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, 0, 0, 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, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 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, 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, 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,
        0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 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],
       [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],
       [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, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 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, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 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, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 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]], dtype=int64)
```

Accuracy of the data

```
In [108]: from sklearn.metrics import accuracy_score
          accuracy_score(y_test, y_pred)

Out[108]: 0.25
```