ZALANDO

Real Discount Rate

Zalando API

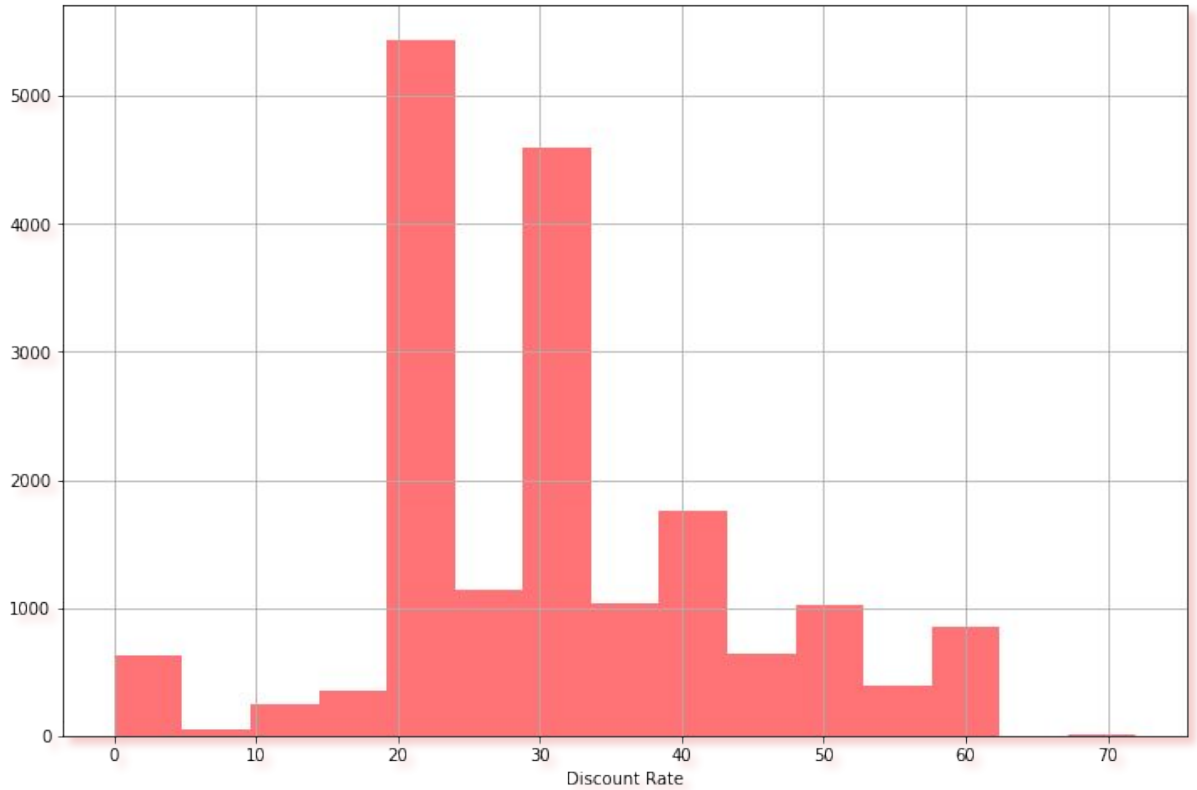
Data Cleaning

sku	object
brand_name	object
flags	object
is_premium	bool
media	object
name	object
price.has_different_original_prices	bool
price.has_different_prices	bool
price.has_different_promotional_prices	bool
price.has_discount_on_selected_sizes_only	bool
price.original	float64
price.promotional	float64
product_group	object
sizes	object
discountRate	object
almost	bool
disc_value	float64
true_discount	float64
name1	object
color	object
category	object
new_category	object
new_color	object

sku	object
is_premium	int64
price.original	float64
discountRate	object
almost	int64
true_discount	float64
tracking_discount	int64
sponsored	int64
10_extra	int64
discount_flag	int64
HOTDROP	int64
tracking_sustainable	int64
new_category_baskets	uint8
new_category_chaussures	uint8
new_category_jean	uint8
new_category_pantalon	uint8
new_category_pyjama	uint8
new_category_robe	uint8
new_category_sandal	uint8
new_category_shirt	uint8
new_category_short	uint8
new_category_sweatshirt	uint8

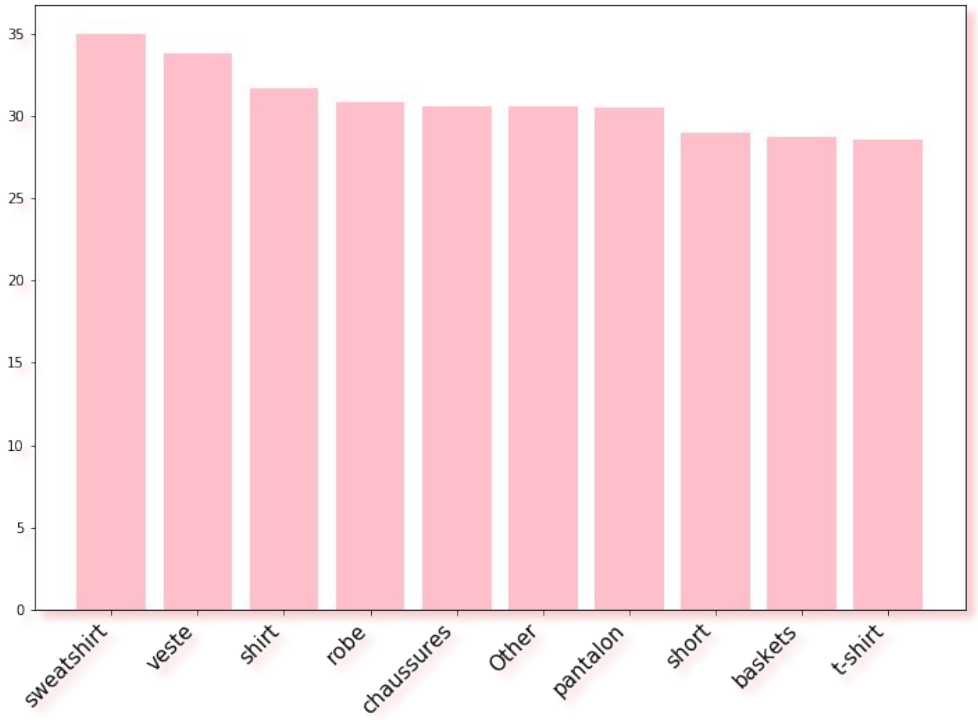
new_category_t-shirt	uint8
new_category_veste	uint8
product_group_beach_wear	uint8
product_group_clothing	uint8
product_group_equipment	uint8
product_group_nightwear	uint8
product_group_shoe	uint8
product_group_underwear	uint8
new_color_black	uint8
new_color_blue	uint8
new_color_green	uint8
new_color_grey	uint8
new_color_navy	uint8
new_color_pink	uint8
new_color_red	uint8
new_color_rose	uint8
new_color_white	uint8
new_color_yellow	uint8

Discount Rate Distribution

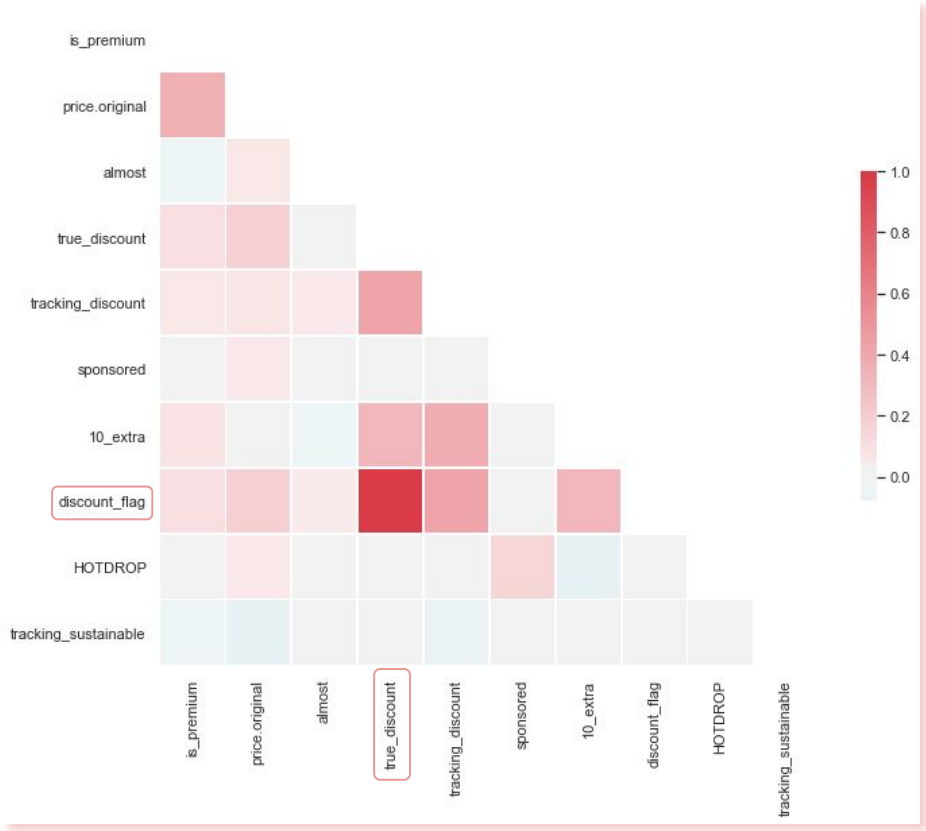


Discount Rate Mean

by Product Type



Correlations




```
zalando.discount_flag.corr(zalando.true_discount)
```

```
0.996483009158388
```

Clustering Data

Drop the most significant columns

```
y=df.discount_flag
```

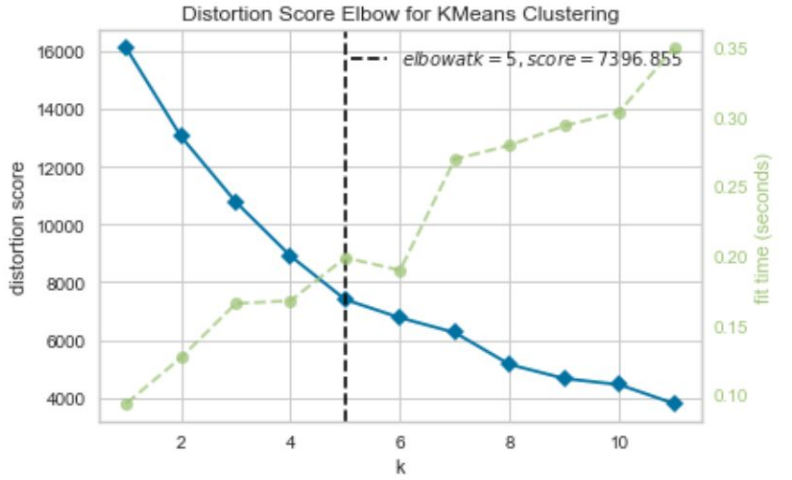
```
X=df.drop(['price.original','almost','discount_flag','product_group_beach_wear',
           'product_group_clothing','product_group_equipment', 'product_group_nightwear',
           'product_group_shoe', 'product_group_underwear','new_color_black','new_color_navy'],
          axis=1)
```

Clustering Data

Choosing the number of clusters

```
model=KMeans()
```

```
visualizer=KElbowVisualizer(model,k=(1,12))
visualizer.fit(X)
visualizer.poof();
```



Clustering Data

1 - Original Data

```
kmeans=KMeans(5)
```

```
df_cluster=kmeans.fit(X)
```

```
df_cluster.cluster_centers_
```

```
df_ml1=df.copy()
```

```
df_ml1['clusters']=df_cluster.labels_
```

```
df_ml1.clusters.value_counts()
```

```
1    7850
0    3009
2    2833
3    2528
4    1911
Name: clusters, dtype: int64
```

2 - Min Max Scaler

```
scaler = MinMaxScaler()
X_sc = scaler.fit_transform(X)
X_sc = pd.DataFrame(X_sc)
```

```
df_cluster2=kmeans.fit(X_sc)
```

```
df_cluster2.cluster_centers_
```

```
df_ml2=df.copy()
df_ml2['clusters']=df_cluster2.labels_
```

```
df_ml2.clusters.value_counts()
```

```
0    8534
1    3353
3    2903
4    1874
2    1467
Name: clusters, dtype: int64
```

3 - PCA & Min Max

```
df_cluster3=kmeans.fit(pca_df)
```

```
df_ml3=df.copy()
```

```
df_ml3['clusters']=df_cluster3.labels_
```

```
df_ml3.clusters.value_counts()
```

```
0    7850
2    3009
3    2833
1    2528
4    1911
Name: clusters, dtype: int64
```

4 - PCA & Original Data

```
df_cluster4=kmeans.fit(pca_df2)
```

```
df_ml4=df.copy()
```

```
df_ml4['clusters']=df_cluster4.labels_
```

Clustering Data

Silhouette Score / Davies Bouldin Score

	Original Data	MinMaxScaler	PCA	PCA + MinMax
3 Clusters	0.38 / 1.26	0.38 / 1.26	0.45 / 1.10	0.43 / 1.14
4 Clusters	0.42 / 1.19	0.43 / 1.21	0.50 / 0.96	0.51 / 0.95
5 Clusters	0.47 / 1.45	0.46 / 1.28	0.58 / 0.79	0.57 / 0.77
6 Clusters	0.52 / 0.97	0.51 / 0.94	0.63 / 0.72	0.62 / 0.74

Machine Learning - Splitting Data

Train on “full discount rates” & testing on the “almost”

- OLS,
- Linear Regression,
- Ridge,
- Lasso,
- ElasticNet

```
X=df[df.almost==0].drop(['almost','discount_flag'],axis=1)
X_2=df[df.almost==1].drop(['almost','discount_flag'],axis=1)
y=df.discount_flag[df.almost==0]
y_2=df.discount_flag[df.almost==1]
```

Machine Learning - Comparing Models

Worse model : OLS

```
pred1 = result1.predict(X1)
```

```
comp1=pd.DataFrame(pred1**4,y_2).reset_index()
comp1['diff']=comp.discount_flag-comp[0]
```

```
comp1.describe()
```

	discount_flag	0	diff
count	1190.000000	977.000000	1190.000000
mean	32.332773	30.985592	3.149192
std	11.400877	3.230710	11.133983
min	5.000000	21.234697	-30.041441
25%	25.000000	29.358460	-4.980805
50%	30.000000	30.326204	1.840535
75%	40.000000	34.342806	9.763058
max	72.000000	38.369490	44.351838

Best model : Ridge **(¼)

```
model3.score(X2,(y**(1/4)))
```

```
0.8173307611153731
```

```
comp3=pd.DataFrame(y_test_pred3**4,y_2).reset_index()
comp3['diff']=comp3.discount_flag-comp3[0]
```

```
comp3.describe()
```

	discount_flag	0	diff
count	1190.000000	1190.000000	1190.000000
mean	32.332773	25.340509	6.992264
std	11.400877	3.140653	11.160077
min	5.000000	16.332787	-30.104634
25%	25.000000	23.456409	-1.318537
50%	30.000000	25.556149	5.545611
75%	40.000000	26.949266	13.674609
max	72.000000	50.104634	47.508463