Dic_project

October 8, 2024

1 Title: Chemical impact evaluation to identify beneficial and harmful ingredients for skin care Products

Problem Statement: • It is competitive for each manufacturers and customers to understand the links among product ingredients, their chemical composition, and customer options in contemporary competitive cosmetics enterprise. • In order to perceive tendencies in component usage, pricing, and brand positioning, this venture will perform an intensive look at of records related to beauty products. • Through the identification of trends in product gives and customer demand, the results can assist groups optimize their advertising procedures and product compositions. • This contribution is crucial to the cosmetic industry's efforts to make certain protection, force innovation in product improvement, and enhance customer's happiness.

Hypothesis questions by Team member-1 : Anirudh 1.Question 1: There Exists outside variables, such advertising and marketing campaigns or emblem popularity, that have an effect on a product's rating without regard to charge?

Objective: To determine whether variables other than price affect the degree of popularity or status of the product. Significance: It may be easier to differentiate between recognition-pushed and price-pushed rankings.

2.Question 3: How a lot of the chemicals within the dataset are classified harmful, and how to use of those chemical compounds evolved through the years?

Objective: To trace the historic use of dangerous materials and check if their availability is diminishing. Significance: For industrial stakeholders and regulatory companies, this may provide important information regarding modifications in chemical safety.

Hypothesis questions by Team member-2 : Rachana Dharmavaram

3.Question 2: Which particular chemical categories are determined in makeup products extra frequently than in other categories of cosmetics?

Objective: To research more about the types of chemicals which might be extra generally discovered in makeup rather than skin care or hair care products. Significance: Finding these substances may result in progressed formulating strategies or legal guidelines.

4.Question 6: Are products designed for certain skin kind like sensitive skin, much more likely to pass over potentially harmful chemicals?

Objective: To discover if sensitive skin products avoid chemical compounds that may reason irritation or aspect effects. Significance: This could offer insights into how the industry adaptive.

Hypothesis questions by Team Member-3 : Satya vaishnavi Jami

5.Question 4: Is there a relationship among a product's popularity or consumer rating and the amount of compounds with few destructive outcomes?

Objective: To inspect if products with much less harmful side effects receive better customer feedback. Significance: This could show how consumers want safer items, which would effect the development of new product designs.

6.Question 5: Do clients price products with color-adding chemical substances decrease or better than the ones without them?

Objective: To inspect if the color-enhancing products impact consumer satisfaction. Significance: Understanding customer's behavior towards color components can inform product development strategies.

[5]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
```

DATASET URLS FROM WHICH WE EXTRACT :- 1)https://www.kaggle.com/datasets/kingabzpro/cosmetics-datasets 2)https://catalog.data.gov/dataset/chemicals-in-cosmetics-2a971/resource/6e8f6b14-040b-40be-a740-a39bb26efbfa

[6]:
```
data=pd.read_csv('merged_table.csv')
```

```
<ipython-input-6-7f8b903e090f>:1: DtypeWarning: Columns (17,20) have mixed
types. Specify dtype option on import or set low_memory=False.
  data=pd.read_csv('merged_table.csv')
```

[7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53763 entries, 0 to 53762
Data columns (total 33 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   CDPHId                 53763 non-null  int64
 1   ProductName            53763 non-null  object
 2   CSFId                  47183 non-null  float64
 3   CSF                    47183 non-null  object
 4   CompanyId              53763 non-null  int64
 5   CompanyName            53763 non-null  object
 6   BrandName              53763 non-null  object
 7   PrimaryCategoryId      53763 non-null  int64
 8   PrimaryCategory        53763 non-null  object
 9   SubCategoryId          53763 non-null  int64
 10  SubCategory            53763 non-null  object
 11  CasId                  53763 non-null  int64
 12  CasNumber              53725 non-null  object
 13  ChemicalId             53763 non-null  int64
 14  ChemicalName           53763 non-null  object
 15  InitialDateReported    53763 non-null  object
 16  MostRecentDateReported 53763 non-null  object
```

```
 17  DiscontinuedDate        7302 non-null   object
 18  ChemicalCreatedAt       53763 non-null  object
 19  ChemicalUpdatedAt       53763 non-null  object
 20  ChemicalDateRemoved     463 non-null    object
 21  ChemicalCount           53763 non-null  int64
 22  Label                   53763 non-null  object
 23  Brand                   53763 non-null  object
 24  Name                    53763 non-null  object
 25  Price                   53763 non-null  int64
 26  Rank                    53763 non-null  float64
 27  Ingredients             53763 non-null  object
 28  Combination             53763 non-null  int64
 29  Dry                     53763 non-null  int64
 30  Normal                  53763 non-null  int64
 31  Oily                    53763 non-null  int64
 32  Sensitive               53763 non-null  int64
dtypes: float64(2), int64(13), object(18)
memory usage: 13.5+ MB
```

Handling Missing Values

Rows with missing values in the CSFId or CSF columns had been removed. This guarantees that null values for categorical variables or key identifiers are , considering that their absence can lead to inconsistent effects in the take a look at.

[8]:
```
data= data.dropna(subset=['CSFId', 'CSF'])
```

In order to put up simplest the columns with missing values, this step counts the entire variety of lacking values for each column. It assists in figuring out the columns that want greater care all through the cleaning process.

[9]:
```
missing_values = data.isnull().sum()
print(missing_values[missing_values>0])
```

```
CasNumber                 32
DiscontinuedDate       41315
ChemicalDateRemoved    46833
dtype: int64
```

Incomplete values in the Chemical Date and Discontinued Date A default price of zero turned into used to fill the removed columns. In addition to making sure that no data is overlooked, this possibly implies that if these dates are missing, the product or chemical hasn't been eliminated.

[10]:
```
data['DiscontinuedDate'].fillna(0,inplace=True)
data['ChemicalDateRemoved'].fillna(0,inplace=True)
```

```
<ipython-input-10-ea265f619f05>:1: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
```

```
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  data['DiscontinuedDate'].fillna(0,inplace=True)
<ipython-input-10-ea265f619f05>:2: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  data['ChemicalDateRemoved'].fillna(0,inplace=True)
```

Converting date columns to datetime

Here by using pd.to_datetime converting all the date columns to perfect temporal data.

[11]:
```
date_col = ['InitialDateReported','DiscontinuedDate', 'ChemicalDateRemoved',␣
↪'MostRecentDateReported', 'ChemicalCreatedAt', 'ChemicalUpdatedAt']
data[date_col] = data[date_col].apply(pd.to_datetime, errors='coerce')
```

Handling categorical data

In handling categorical data, we took category columns and found many distinct values that is why we used high cardinality and for repeated values we used low cardinality. And then removed all the dummies in the data using low cardinality

[12]:
```
cat_cols= ['ProductName', 'CSF', 'CompanyName', 'BrandName',
    'PrimaryCategory', 'SubCategory', 'CasNumber',
    'ChemicalName', 'Label', 'Brand', 'Name', 'Ingredients']
```

[13]:
```
high_cardinality_cols = []
for col in cat_cols:
    if data[col].nunique() > 100:
        high_cardinality_cols.append(col)
```

[14]:
```
low_cardinality_cols = []
for col in cat_cols:
    if data[col].nunique() <= 100:
        low_cardinality_cols.append(col)
```

[15]:
```
data= pd.get_dummies(data, columns=low_cardinality_cols)
```

[16]:
```
from sklearn.preprocessing import LabelEncoder
```

[17]:
```
encoder=LabelEncoder()
for col in high_cardinality_cols:
    data[col]=encoder.fit_transform(data[col])
```

Feature Scaling

To normalize the range of independent variables and the features of data we used StandardScaler() method.

[18]:
```
number_cols = ['CDPHId', 'CSFId', 'CompanyId', 'PrimaryCategoryId',␣
↪'SubCategoryId', 'CasId',
               'ChemicalId', 'ChemicalCount', 'Price',
               'Rank', 'Combination', 'Dry', 'Normal', 'Oily', 'Sensitive']
```

[19]:
```
scaler=StandardScaler()
data[number_cols]=scaler.fit_transform(data[number_cols])
```

Removing duplicates in the data

This removes duplicate entries so that each data point is unique and to avoid skewed analysis.

[20]:
```
data=data.drop_duplicates()
```

Encoding dates

[21]:
```
date_columns=['InitialDateReported', 'MostRecentDateReported',
                'DiscontinuedDate', 'ChemicalCreatedAt',
                'ChemicalUpdatedAt']
```

[22]:
```
for col in date_columns:
    data[col + '_year'] = data[col].dt.year
    data[col + '_month'] = data[col].dt.month
    data[col + '_day'] = data[col].dt.day
```

[23]:
```
data.drop(columns=date_columns, inplace=True)
```

Handling outliers

We did this to handle the aqnomalies and to deal with the outliers.

[24]:
```
price_quantiles = data['Price'].quantile([0.01, 0.99])
data = data[(data['Price'] >= price_quantiles[0.01]) & (data['Price'] <=␣
↪price_quantiles[0.99])]
```

[25]:
```
for col in ['Rank', 'Combination', 'Dry', 'Normal', 'Oily', 'Sensitive']:
    data[col] = data[col].astype(int)
    upper= data[col].quantile(0.99)
```

```
    data=data[data[col]<=upper]
    lower=data[col].quantile(0.01)
    data=data[data[col]>=lower]
```

```
<ipython-input-25-5fec5752f02f>:2: SettingWithCopyWarning:
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: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data[col] = data[col].astype(int)
```

Z-Score

Z-score is the mathematical function for evaluation and verification.This helps us to find how far a piece of data from the average of group.

[26]:
```
z_scores = (data['Price'] - data['Price'].mean()) / data['Price'].std()
data = data[(z_scores.abs() < 3)]
```

[27]:
```
print(z_scores)
```

```
0       -0.730867
1       -0.114872
2       -0.357537
3       -0.525535
238     -0.656201
           …
53598   -0.730867
53599   -0.637535
53600   -0.730867
53601   -0.637535
53602   -0.730867
Name: Price, Length: 46592, dtype: float64
```

[28]:
```
print(data.describe())
```

```
             CDPHId   ProductName         CSFId            CSF     CompanyId  \
count  44519.000000  44519.000000  44519.000000  44519.000000  44519.000000
mean      -0.002522    250.313686      0.001532   1564.302478      0.001577
min       -3.057504      0.000000     -2.970937      0.000000     -1.892778
25%       -0.031196    197.000000     -0.138737    812.000000     -1.030252
50%        0.228121    302.000000      0.114082   1535.000000      0.692185
75%        0.404921    311.000000      0.287787   2252.000000      0.692185
max        2.529831    392.000000      2.685346   3081.000000      1.267202
std        1.018283     93.951001      1.017473    838.511810      1.001821

       PrimaryCategoryId  SubCategoryId         CasId    ChemicalId  \
count       44519.000000   44519.000000  44519.000000  44519.000000
mean           -0.001771      -0.003258     -0.013202     -0.000996
```

```
min            -0.995989      -1.109041     -5.928370     -2.881365
25%            -0.507629      -0.511736     -0.116705     -0.168526
50%            -0.507629      -0.511736     -0.116705      0.124070
75%            -0.507629      -0.383742     -0.116705      0.377788
max             2.015567       4.266704      4.499653      2.674413
std             0.998463       0.997922      0.995074      1.019475

                ChemicalDateRemoved  …  MostRecentDateReported_day  \
count                         44519  …                44519.000000
mean  1970-04-25 06:19:19.109593656  …                   14.072890
min             1970-01-01 00:00:00  …                    1.000000
25%             1970-01-01 00:00:00  …                    1.000000
50%             1970-01-01 00:00:00  …                   13.000000
75%             1970-01-01 00:00:00  …                   25.000000
max             2016-08-29 00:00:00  …                   31.000000
std                             NaN  …                   10.852228

      DiscontinuedDate_year  DiscontinuedDate_month  DiscontinuedDate_day  \
count           5363.000000             5363.000000           5363.000000
mean            2014.027224                3.156256              6.153832
min             2008.000000                1.000000              1.000000
25%             2014.000000                1.000000              1.000000
50%             2014.000000                1.000000              1.000000
75%             2014.000000                7.000000              1.000000
max             2018.000000               12.000000             31.000000
std                1.637063                3.476544             10.951966

      ChemicalCreatedAt_year  ChemicalCreatedAt_month  ChemicalCreatedAt_day  \
count           44519.000000             44519.000000           44519.000000
mean             2014.193288                 5.753835              20.153081
min              2009.000000                 1.000000               1.000000
25%              2014.000000                 5.000000              19.000000
50%              2015.000000                 5.000000              20.000000
75%              2015.000000                 8.000000              25.000000
max              2019.000000                12.000000              31.000000
std                 2.330058                 3.076650               7.593042

      ChemicalUpdatedAt_year  ChemicalUpdatedAt_month  ChemicalUpdatedAt_day
count           44519.000000             44519.000000           44519.000000
mean             2014.862935                 6.026011              17.711606
min              2010.000000                 1.000000               1.000000
25%              2014.000000                 5.000000              11.000000
50%              2015.000000                 5.000000              19.000000
75%              2015.000000                 8.000000              23.000000
max              2019.000000                12.000000              31.000000
std                 1.483391                 3.172844               7.602613

[8 rows x 35 columns]
```

[28]:

```

```