```
# This Python 3 environment comes with many helpful analytics
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing Shift+Enter)
will list all files under the input directory



# You can write up to 20GB to the current directory (/kaggle/working/)
that gets preserved as output when you create a version using "Save &
Run All"
# You can also write temporary files to /kaggle/temp/, but they won't
be saved outside of the current session
```

H&M Personalized Fashion Recommendation

Data Set:

1. images.csv: Product images
2. articles.csv: Detailed data about each product
3. customers.csv: Detailed infomation about each customer
4. transactions.csv: Detailed data about each purchase

```
# loading data
import pandas as pd
articles = pd.read_csv('../input/h-and-m-personalized-fashion-
recommendations/articles.csv')
customers = pd.read_csv('../input/h-and-m-personalized-fashion-
recommendations/customers.csv')
transactions = pd.read_csv('../input/h-and-m-personalized-fashion-
recommendations/transactions_train.csv',dtype={'article_id': str})

print(articles.columns)
print(customers.columns)
print(transactions.columns)
```

Transaction table is our training table, with columns article_id and customer_id as foreign keys to articles and customers tables.

```
print(articles.info())
print(customers.info())
print(transactions.info())
```

Preview missing data and unique data

```
def missing_data(data):
    total = data.isnull().sum().sort_values(ascending = False)
    percent =
(data.isnull().sum()/data.isnull().count()*100).sort_values(ascending
= False)
    return pd.concat([total, percent], axis=1, keys=['Total',
'Percent'])

def unique_values(data):
    total = data.count()
    tt = pd.DataFrame(total)
    tt.columns = ['Total']
    uniques = []
    for col in data.columns:
        unique = data[col].nunique()
        uniques.append(unique)
    tt['Uniques'] = uniques
    return tt

missing_data(articles)
```

In articles, only 0.4% values from detail description are missing.

```
missing_data(customers)
```

In table customers, customer_id and postal_code are completely filled, whereas around 1% data of age, fashion_news_frequency are missing. And a high percent of 66% missing data from Active stuts and FN.

```
missing_data(transactions)

unique_values(articles)

unique_values(customers)

unique_values(transactions)

print(f"Percent of articles present in transactions:
{round(104547/105542,3)*100}%")
print(f"Percent of customer present in transactions:
{round(1362281/1371980,3)*100}%")
```

Articles Data Visualization

```
import seaborn as sns
import matplotlib.pyplot as plt
```

Number of **products types** per each **Product Group**

```
types_group = articles.groupby(["product_group_name"])
["product_type_name"].nunique()

df = pd.DataFrame({'Product Group': types_group.index,
                   'Product Types': types_group.values
                  })
df = df.sort_values(['Product Types'], ascending=False)
print(types_group)

plt.figure(figsize = (8,6))
plt.title('Number of Product Types per each Product Group')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Product Group', y="Product Types", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()


from wordcloud import WordCloud, STOPWORDS
stopwords = set(STOPWORDS)

def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=5,
        random_state=1
    ).generate(str(data))

    fig = plt.figure(1, figsize=(10,10))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=14)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(articles["prod_name"], "Wordcloud from product name")
```

Number of Articles per each **product group**

```
articles_group = articles.groupby(['product_group_name'])
['article_id'].nunique()

df = pd.DataFrame({'Product Group':articles_group.index, 'Number of
```

```
articles':articles_group.values}).sort_values(by='Number of articles',
ascending=False)
plt.figure(figsize = (6,4))
plt.title('Number of articles per product group')
s = sns.barplot(x='Product Group', y='Number of articles', data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```

NUmber of articles per each **product type**

```
article_type = articles.groupby(['product_type_name'])
['article_id'].nunique()
df = pd.DataFrame({'Product Type Name':article_type.index, 'Number of
Articles':article_type.values}).sort_values(by='Number of Articles',
ascending=False)
plt.figure(figsize = (20,4))
plt.title('Number of articles per product type')
s = sns.barplot(x='Product Type Name', y='Number of Articles',
data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```

Number of articles per department

```
article_department = articles.groupby(['department_name'])
['article_id'].nunique()
df = pd.DataFrame({'Department Name':article_department.index, 'Number
of articles':article_department.values}).sort_values(by='Number of
articles', ascending=False).head(50)
plt.figure(figsize = (50,4))
plt.title('Number of articles per department top 50')
s = sns.barplot(x='Department Name', y='Number of articles', data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```

Number of Articles per each Graphical Appearance Name

```
articles_ganame = articles.groupby(['graphical_appearance_name'])
['article_id'].nunique()
df = pd.DataFrame({'Graphical Appearance Name':articles_ganame.index,
'Number of Articles':articles_ganame.values}).sort_values(by='Number
of Articles', ascending=False).head(10)
plt.figure(figsize = (16, 4))
plt.title('Number of Articles per graphical appearance name')
s = sns.barplot(x='Graphical Appearance Name', y='Number of Articles',
data=df)
plt.show()
```

Number of Articles per each **Index Group** Name

```
article_index = articles.groupby(['index_group_name'])
['article_id'].nunique()
df = pd.DataFrame({'Index Group Name':article_index.index, 'Number of
Articles':article_index.values}).sort_values(by='Number of Articles',
ascending=False)
plt.figure(figsize = (10, 6))
plt.title('Number of articles per each index group')
s = sns.barplot(x='Index Group Name', y='Number of Articles', data=df)
plt.show()
```

Number of Articles per each **Colour Group** Name

```
article_colour = articles.groupby(['colour_group_name'])
['article_id'].nunique()
df = pd.DataFrame({'Colour Group Name': article_colour.index, 'Number
of Articles': article_colour.values}).sort_values(by='Number of
Articles', ascending= False).head(15)
plt.figure(figsize=(20, 5))
plt.title('Number of Articles per each colour group name top 15')
s = sns.barplot(x='Colour Group Name', y='Number of Articles',
data=df)
plt.show()
```

Number of Articles per each **Perceived Colour Group** Name

```
article_pcolour = articles.groupby(['perceived_colour_value_name'])
['article_id'].nunique()
df = pd.DataFrame({'Perceived Colour Group Name':
article_pcolour.index, 'Number of Articles':
article_pcolour.values}).sort_values(by='Number of Articles',
ascending= False)
plt.figure(figsize=(20, 5))
plt.title('Number of Articles per each perceived colour group name')
s = sns.barplot(x='Perceived Colour Group Name', y='Number of
Articles', data=df)
plt.show()
```

Number of Articles per each **Perceived Colour Master Name**

```
article_pmcolour = articles.groupby(['perceived_colour_master_name'])
['article_id'].nunique()
df = pd.DataFrame({'Perceived Colour Master Name':
article_pmcolour.index, 'Number of Articles':
article_pmcolour.values}).sort_values(by='Number of Articles',
ascending= False)
plt.figure(figsize=(20, 5))
plt.title('Number of Articles per each Perceived Colour Master Name')
s = sns.barplot(x='Perceived Colour Master Name', y='Number of
Articles', data=df)
plt.show()
```

Number of Articles per each**Index Name**

```
article_index = articles.groupby(['index_name'])
['article_id'].nunique()
df = pd.DataFrame({'Index Name': article_index.index, 'Number of
Articles': article_index.values}).sort_values(by='Number of Articles',
ascending= False)
plt.figure(figsize=(20, 5))
plt.title('Number of Articles per each Index Name')
s = sns.barplot(x='Index Name', y='Number of Articles', data=df)
plt.show()
```

Number of Articles per each **Garment Group** Name

```
article_index = articles.groupby(['garment_group_name'])
['article_id'].nunique()
df = pd.DataFrame({'Garment Group Name': article_index.index, 'Number
of Articles': article_index.values}).sort_values(by='Number of
Articles', ascending= False)
plt.figure(figsize=(20, 5))
plt.title('Number of Articles per each Garment Group Name')
s = sns.barplot(x='Garment Group Name', y='Number of Articles',
data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
plt.show()
```

Number of Articles per each **Section Name**

```
article_index = articles.groupby(['section_name'])
['article_id'].nunique()
df = pd.DataFrame({'Section Name': article_index.index, 'Number of
Articles': article_index.values}).sort_values(by='Number of Articles',
ascending= False)
plt.figure(figsize=(20, 5))
plt.title('Number of Articles per each Section Name')
s = sns.barplot(x='Section Name', y='Number of Articles', data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
plt.show()
```

word cloud for detail description

```
show_wordcloud(articles["detail_desc"], "Wordcloud from detailed
description of articles")
```

Customers data

Number of Customers per each age

```
temp = customers.groupby(["age"])["customer_id"].count()
df = pd.DataFrame({'Age': temp.index,
                   'Customers': temp.values
                  })
df = df.sort_values(['Age'], ascending=False)
plt.figure(figsize = (16,6))
```

```
plt.title(f'Number of Customers per each Age')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Age', y="Customers", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```

Number of Customers per each Fashion News Frequency

```
temp = customers.groupby(["fashion_news_frequency"])
["customer_id"].count()
df = pd.DataFrame({'Fashion News Frequency': temp.index,
                   'Customers': temp.values
                  })
df = df.sort_values(['Customers'], ascending=False)
plt.figure(figsize = (6,6))
plt.title(f'Number of Customers per each Fashion News Frequency')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Fashion News Frequency', y="Customers", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```

Number of customers per each activity status

```
temp = customers.groupby(["club_member_status"])
["customer_id"].count()
df = pd.DataFrame({'Club Member Status': temp.index,
                   'Customers': temp.values
                  })
df = df.sort_values(['Customers'], ascending=False)
plt.figure(figsize = (6,6))
plt.title(f'Number of Customers per each Club Member Status')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Club Member Status', y="Customers", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```

Transaction data

```
import numpy as np
df = transactions.sample(100_000)
fig, ax = plt.subplots(1, 1, figsize=(14, 7))
sns.kdeplot(np.log(df.loc[df["sales_channel_id"]==1].price.value_count
s()))
sns.kdeplot(np.log(df.loc[df["sales_channel_id"]==2].price.value_count
s()))
ax.legend(labels=['Sales channel 1', 'Sales channel 1'])
plt.title("Logaritmic distribution of price frequency in transactions,
```

```
grouped per sales channel (100k sample)")
plt.show()

print(transactions.shape)
transactions.sort_values(by='t_dat', ascending=False).head()

transactions['t_dat'] = pd.to_datetime(transactions['t_dat'])

transactions_3w = transactions[transactions['t_dat'] >=
pd.to_datetime('2020-08-31')].copy()
transactions_2w = transactions[transactions['t_dat'] >=
pd.to_datetime('2020-09-07')].copy()
transactions_1w = transactions[transactions['t_dat'] >=
pd.to_datetime('2020-09-15')].copy()

purchase_dict_3w = {}

for i,x in enumerate(zip(transactions_3w['customer_id'],
transactions_3w['article_id'])):
    cust_id, art_id = x
    if cust_id not in purchase_dict_3w:
        purchase_dict_3w[cust_id] = {}

    if art_id not in purchase_dict_3w[cust_id]:
        purchase_dict_3w[cust_id][art_id] = 0

    purchase_dict_3w[cust_id][art_id] += 1

print(len(purchase_dict_3w))

dummy_list_3w =
list((transactions_3w['article_id'].value_counts()).index)[:12]

purchase_dict_2w = {}

for i,x in enumerate(zip(transactions_2w['customer_id'],
transactions_2w['article_id'])):
    cust_id, art_id = x
    if cust_id not in purchase_dict_2w:
        purchase_dict_2w[cust_id] = {}

    if art_id not in purchase_dict_2w[cust_id]:
        purchase_dict_2w[cust_id][art_id] = 0

    purchase_dict_2w[cust_id][art_id] += 1

print(len(purchase_dict_2w))

dummy_list_2w =
list((transactions_2w['article_id'].value_counts()).index)[:12]
```

```
purchase_dict_1w = {}

for i,x in enumerate(zip(transactions_1w['customer_id'],
transactions_1w['article_id'])):
    cust_id, art_id = x
    if cust_id not in purchase_dict_1w:
        purchase_dict_1w[cust_id] = {}

    if art_id not in purchase_dict_1w[cust_id]:
        purchase_dict_1w[cust_id][art_id] = 0

    purchase_dict_1w[cust_id][art_id] += 1

print(len(purchase_dict_1w))

dummy_list_1w =
list((transactions_1w['article_id'].value_counts()).index)[:12]
#purchase_dict_1w

submission = pd.read_csv('../input/h-and-m-personalized-fashion-
recommendations/sample_submission.csv')
print(submission.shape)
submission.head()

prediction = submission[['customer_id']]
prediction_list = []

dummy_list =
list((transactions_1w['article_id'].value_counts()).index)[:12]
dummy_pred = ' '.join(dummy_list)

for i, cust_id in
enumerate(submission['customer_id'].values.reshape((-1,))):
    if cust_id in purchase_dict_1w:
        l = sorted((purchase_dict_1w[cust_id]).items(), key=lambda x:
x[1], reverse=True)
        l = [y[0] for y in l]
        if len(l)>12:
            s = ' '.join(l[:12])
        else:
            s = ' '.join(l+dummy_list_1w[:(12-len(l))])
    elif cust_id in purchase_dict_2w:
        l = sorted((purchase_dict_2w[cust_id]).items(), key=lambda x:
x[1], reverse=True)
        l = [y[0] for y in l]
        if len(l)>12:
            s = ' '.join(l[:12])
        else:
            s = ' '.join(l+dummy_list_2w[:(12-len(l))])
    elif cust_id in purchase_dict_3w:
```

```
        l = sorted((purchase_dict_3w[cust_id]).items(), key=lambda x:
x[1], reverse=True)
        l = [y[0] for y in l]
        if len(l)>12:
            s = ' '.join(l[:12])
        else:
            s = ' '.join(l+dummy_list_3w[:(12-len(l))])
    else:
        s = dummy_pred
    prediction_list.append(s)

prediction['prediction'] = prediction_list
print(prediction.shape)
prediction.head()

prediction.to_csv('prediction.csv', index=False)
```