In [1]:
```
#import all packages
import os
import numpy as np
import pandas as pd
import cv2 as cv
from pathlib import Path
import warnings
from skimage.feature import hog
import tqdm

warnings.filterwarnings("ignore")
pd.options.display.max_columns = None
```

In [2]:
```
#define paths and read data
image_dir = 'e:/dev/Kaggle/fashion/'
style_file = 'styles.csv'
image_folder = image_dir + '/images/'

#load image
styles = pd.read_csv(Path(image_dir+style_file),error_bad_lines=False)
```

```
b'Skipping line 6044: expected 10 fields, saw 11\nSkipping line 6569: expected
10 fields, saw 11\nSkipping line 7399: expected 10 fields, saw 11\nSkipping lin
e 7939: expected 10 fields, saw 11\nSkipping line 9026: expected 10 fields, saw
11\nSkipping line 10264: expected 10 fields, saw 11\nSkipping line 10427: expec
ted 10 fields, saw 11\nSkipping line 10905: expected 10 fields, saw 11\nSkippin
g line 11373: expected 10 fields, saw 11\nSkipping line 11945: expected 10 fiel
ds, saw 11\nSkipping line 14112: expected 10 fields, saw 11\nSkipping line 1453
2: expected 10 fields, saw 11\nSkipping line 15076: expected 10 fields, saw 12
\nSkipping line 29906: expected 10 fields, saw 11\nSkipping line 31625: expecte
d 10 fields, saw 11\nSkipping line 33020: expected 10 fields, saw 11\nSkipping
line 35748: expected 10 fields, saw 11\nSkipping line 35962: expected 10 field
s, saw 11\nSkipping line 37770: expected 10 fields, saw 11\nSkipping line 3810
5: expected 10 fields, saw 11\nSkipping line 38275: expected 10 fields, saw 11
\nSkipping line 38404: expected 10 fields, saw 12\n'
```

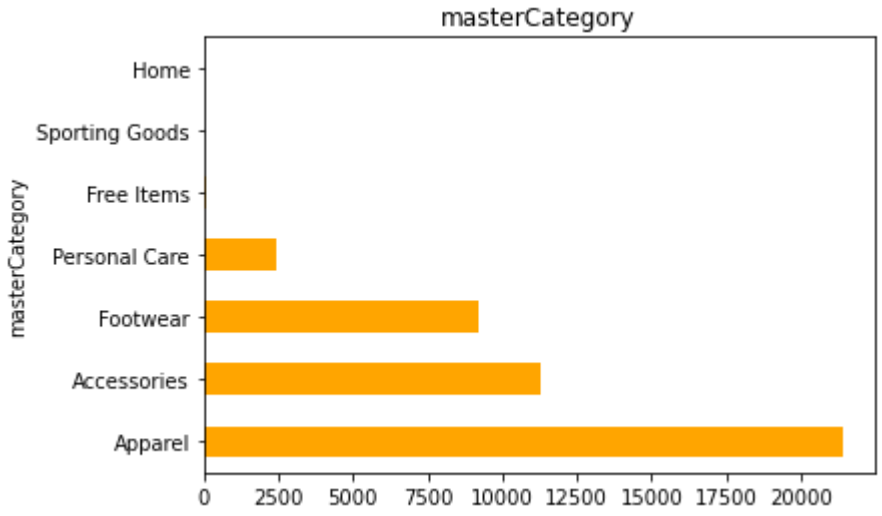
In [3]:
```
#style file
print("Style shape: ", str(styles.shape))
styles.head(10)
```

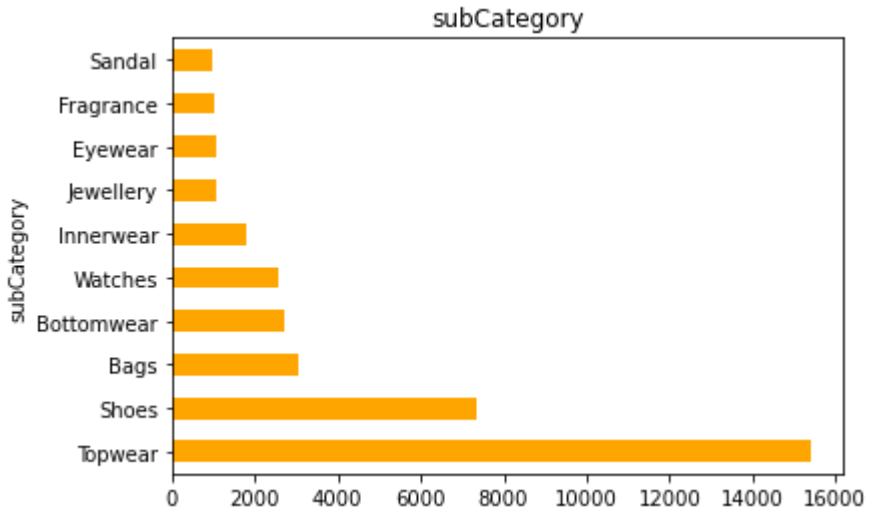
```
Style shape:  (44424, 10)
```

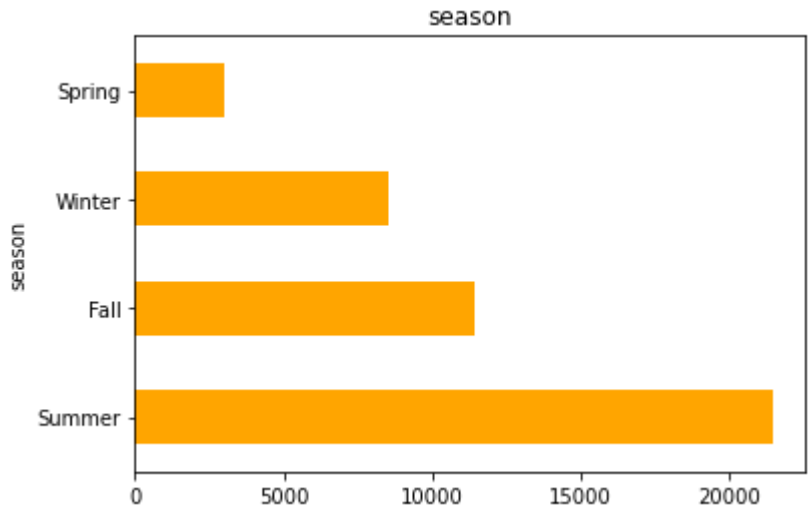
Out[3]:

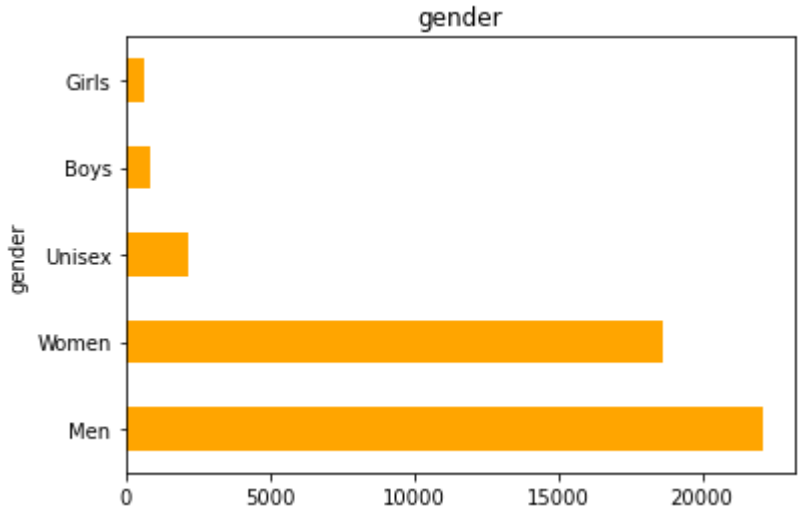
	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage
0	15970	Men	Apparel	Topwear	Shirts	Navy Blue	Fall	2011.0	Casual
1	39386	Men	Apparel	Bottomwear	Jeans	Blue	Summer	2012.0	Casual
2	59263	Women	Accessories	Watches	Watches	Silver	Winter	2016.0	Casual
3	21379	Men	Apparel	Bottomwear	Track Pants	Black	Fall	2011.0	Casual
4	53759	Men	Apparel	Topwear	Tshirts	Grey	Summer	2012.0	Casual
5	1855	Men	Apparel	Topwear	Tshirts	Grey	Summer	2011.0	Casual
6	30805	Men	Apparel	Topwear	Shirts	Green	Summer	2012.0	Ethnic
7	26960	Women	Apparel	Topwear	Shirts	Purple	Summer	2012.0	Casual
8	29114	Men	Accessories	Socks	Socks	Navy Blue	Summer	2012.0	Casual
9	30039	Men	Accessories	Watches	Watches	Black	Winter	2016.0	Casual

In [4]:
```
#basic EDA, check bucket and bins across major categories
import matplotlib.pyplot as plt
from tqdm import tqdm
categories = ['masterCategory','subCategory','season','gender' ]

#plot above five class vars
for cat in categories:
    df_cat = styles.groupby(cat,as_index=False).size().sort_values(ascending=Fals
    df_cat.plot(kind='barh',title = cat, color="orange")
    plt.show()
```









In [5]:
```
all_images = []


def load_image(ids,path=image_folder):
    #Load gray scale imges
    img = cv.imread(image_folder+ids+'.jpg',cv.IMREAD_GRAYSCALE)
    return img,ids

for ids in tqdm(list(styles.id)[:15000]):
    img,ids = load_image(str(ids))
    if img is not None:
        all_images.append([img,int(ids)])
    #Labels.append(ids)
len(all_images)
```

```
100%|██████████| 15000/15000 [09:35<00:00, 26.08it/s]
```

Out[5]: 14999

In [6]:
```
def resize_image(img,ids):
    return cv.resize(img, (60, 80),interpolation =cv.INTER_LINEAR)

all_images_resized = [[resize_image(x,y),y] for x,y in all_images]
len(all_images_resized)
```

Out[6]: 14999

In [7]:
```
styles.head()
```

Out[7]:

	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage
0	15970	Men	Apparel	Topwear	Shirts	Navy Blue	Fall	2011.0	Casual
1	39386	Men	Apparel	Bottomwear	Jeans	Blue	Summer	2012.0	Casual
2	59263	Women	Accessories	Watches	Watches	Silver	Winter	2016.0	Casual
3	21379	Men	Apparel	Bottomwear	Track Pants	Black	Fall	2011.0	Casual
4	53759	Men	Apparel	Topwear	Tshirts	Grey	Summer	2012.0	Casual

In [8]:
```
[styles.masterCategory.value_counts().index]
```

Out[8]:
```
[Index(['Apparel', 'Accessories', 'Footwear', 'Personal Care', 'Free Items',
        'Sporting Goods', 'Home'],
       dtype='object')]
```

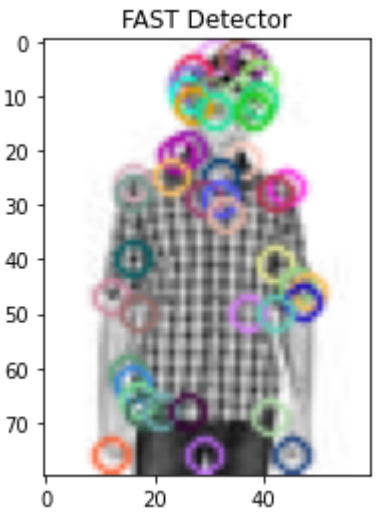
In [9]:
```
df_labels = pd.DataFrame(all_images_resized,columns=['image','id'])

target = 'masterCategory'
categories = ['Apparel', 'Accessories', 'Footwear', 'Personal Care', 'Free Items'
df_train = styles[styles[target].isin(categories)][['id',target]]

df_labels = pd.merge(df_labels,df_train,how='left',on=['id'])
df_labels = df_labels.fillna('Others')
df_labels['class'] = pd.factorize(df_labels[target])[0]
print("Data Shape: ", str(df_labels.shape))
print(df_labels[target].value_counts())
```

```
Data Shape:  (14999, 4)
Apparel          7329
Accessories      3793
Footwear         3021
Personal Care     818
Free Items         30
Others              8
Name: masterCategory, dtype: int64
```

In [10]:
```
#mapper for targets and labels
mapper = df_labels[['class',target]].drop_duplicates()
```

In [11]:
```
for image in df_labels.image[:20]:
    print(image.shape)
    plt.imshow(image)
    fast = cv.FastFeatureDetector_create(50)
    kp = fast.detect(image,None)
    img2 = cv.drawKeypoints(image, kp, None, color=(255,0,0))
    # Print all default params
    #print( "Threshold: {}".format(fast.getThreshold()) )
    #print( "neighborhood: {}".format(fast.getType()) )
    print( "Total Keypoints with nonmaxSuppression: {}".format(len(kp)))
    fast_image=cv.drawKeypoints(image,kp,image)
    plt.imshow(fast_image);plt.title('FAST Detector')
    plt.show()
```

```
(80, 60)
Total Keypoints with nonmaxSuppression: 44
```



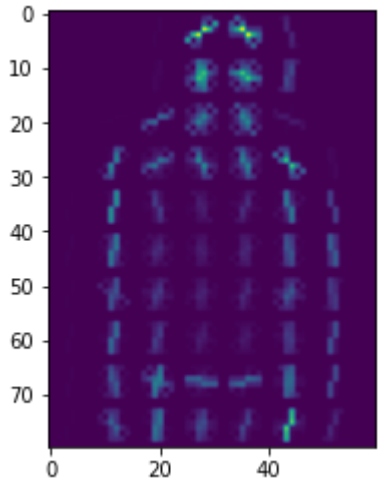
In [12]:
```
train_images = np.stack(df_labels.image.values,axis=0)
n_samples = len(train_images)
data_images = train_images.reshape((n_samples, -1))
```

In [13]:
```
ppcr = 8
ppcc = 8
hog_images = []
hog_features = []
for image in tqdm(train_images):
    blur = cv.GaussianBlur(image,(5,5),0)
    fd,hog_image = hog(blur, orientations=8, pixels_per_cell=(ppcr,ppcc),cells_pe
    hog_images.append(hog_image)
    hog_features.append(fd)

hog_features = np.array(hog_features)

hog_features.shape
```

```
100%|██████████████████████████████| 14999/14999 [02:44<00:00, 91.02it/s]
```

Out[13]:
```
(14999, 1728)
```

In [14]:
```
for img in hog_images[:20]:
    plt.imshow(img)
    plt.show()
```



In [15]:
```
edges = [cv.Canny(image,50,150,apertureSize = 3) for image in train_images]
edges = np.array(edges)
n_samples_edges = len(edges)
edge_images = edges.reshape((n_samples, -1))
edge_images.shape
```

Out[15]:
```
(14999, 4800)
```

In [16]:
```
train_images.shape, hog_features.shape, edge_images.shape
```

Out[16]:
```
((14999, 80, 60), (14999, 1728), (14999, 4800))
```

In [17]:
```
edge_hog = np.hstack([hog_features,edge_images])
edge_hog.shape
```

Out[17]:
```
(14999, 6528)
```

In [18]:
```
histr = [cv.calcHist([img],[0],None,[256],[0,256]) for img in train_images]
histr = np.array(histr)
n_samples_histr = len(histr)
image_hist = histr.reshape((n_samples_histr, -1))
image_hist.shape
```

Out[18]:
```
(14999, 256)
```

In [19]:
```
edge_hog = np.hstack([hog_features,edge_images,image_hist])
edge_hog.shape
```

Out[19]:
```
(14999, 6784)
```

In [20]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(hog_features,df_labels['class
print('Training data and target sizes: \n{}, {}'.format(X_train.shape,y_train.sha
print('Test data and target sizes: \n{}, {}'.format(X_test.shape,y_test.shape))
```

```
Training data and target sizes:
(11999, 1728), (11999,)
Test data and target sizes:
(3000, 1728), (3000,)
```

In [21]:
```
y_train.value_counts(),y_test.value_counts()
```

Out[21]:
```
(0    5863
 1    3034
 2    2417
 3     654
 4      24
 5       7
 Name: class, dtype: int64,
 0    1466
 1     759
 2     604
 3     164
 4       6
 5       1
 Name: class, dtype: int64)
```

In [22]:
```
from sklearn import datasets, svm, metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
# # Create a classifier: a support vector classifier
# classifier = svm.SVC(gamma=0.001)
# #fit to the trainin data
# classifier.fit(X_train,y_train)
```

In [23]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier

test_accuracy = []
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_train)

classifier = KNeighborsClassifier(n_neighbors=3,algorithm='brute')
classifier.fit(X_scaled, y_train)
test_accuracy = classifier.score(scaler.transform(X_test), y_test)
print(test_accuracy)

# #FOR TUNING
# print(search_params)
# for p in tqdm(search_params):
#     #classifier = svm.SVC(gamma=p)
#     classifier = RandomForestClassifier(max_depth=8,n_estimators=600)

#     classifier.fit(X_scaled, y_train)
#     test_accuracy.append([p,classifier.score(scaler.transform(X_test), y_test)]

# df_accuracy = pd.DataFrame(test_accuracy,columns =['gamma','accuracy'])
# df_accuracy.index = df_accuracy.gamma
# df_accuracy[['accuracy']].plot()
# plt.show()
```

```
0.974
```

In [24]:
```
mapper= mapper.reset_index(drop=True)
```

In [25]:
```
y_pred = classifier.predict(scaler.transform(X_test))

df_result = pd.DataFrame(y_test)
df_result['id'] = df_result.index
df_result = df_result.rename(columns={'class':'actual'})
df_result['predicted'] = y_pred
df_result = df_result.reset_index(drop=True)
df_result = pd.merge(df_result,mapper,left_on='predicted',right_on = 'class',how=
df_result = df_result.drop(columns=['class'],axis=1)
df_result = df_result.rename(columns={'gender':'predicted_category'})

df_result = pd.merge(df_result,mapper,left_on='actual',right_on = 'class',how='ir
df_result = df_result.drop(columns=['class'],axis=1)
df_result.shape
```

Out[25]:
```
(3000, 5)
```

In [26]:
```
#some references for debugging
kd = df_result[df_result.actual!=df_result.predicted]
print(kd.shape)
kd.head()
```

```
(78, 5)
```

Out[26]:

	actual	id	predicted	masterCategory_x	masterCategory_y
1451	0	11972	2	Footwear	Apparel
1452	0	13311	2	Footwear	Apparel
1453	0	14873	1	Accessories	Apparel
1454	0	12649	1	Accessories	Apparel
1455	0	9228	1	Accessories	Apparel

In [27]:
```
# image_id = styles[styles.index==2663]['id'].reset_index(drop=True)
# k = str(image_id)
# print(k)
#print(image_folder+str(image_id)+'.jpg')

#debug image with id
##it is recommended not to use the image used for training. You can make a separa
img = cv.imread(image_folder+str(7347)+'.jpg')
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
img.shape
plt.imshow(img)
```

Out[27]:
```
<matplotlib.image.AxesImage at 0x4a3877c0>
```

In [28]:
```
list_of_categories = categories +['Others']

print("Classification Report: \n Target: %s \n Labels: %s \n Classifier: %s:\n%s\
      % (target,list_of_categories,classifier, metrics.classification_report(y_te

df_report = pd.DataFrame(metrics.confusion_matrix(y_test, y_pred),columns = list_
df_report.index = [list_of_categories]
df_report
```

```
Classification Report:
 Target: masterCategory
 Labels: ['Apparel', 'Accessories', 'Footwear', 'Personal Care', 'Free Items',
'Others']
 Classifier: KNeighborsClassifier(algorithm='brute', n_neighbors=3):
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      1466
           1       0.97      0.95      0.96       759
           2       0.99      0.99      0.99       604
           3       0.88      0.93      0.91       164
           4       0.00      0.00      0.00         6
           5       0.00      0.00      0.00         1

    accuracy                           0.97      3000
   macro avg       0.64      0.64      0.64      3000
weighted avg       0.97      0.97      0.97      3000
```

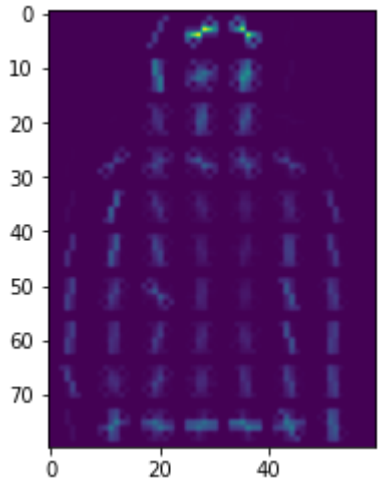
Out[28]:

	Apparel	Accessories	Footwear	Personal Care	Free Items	Others
Apparel	1451	8	2	5	0	0
Accessories	22	722	3	12	0	0
Footwear	2	2	596	4	0	0
Personal Care	2	8	1	153	0	0
Free Items	2	4	0	0	0	0
Others	0	1	0	0	0	0

In [29]:
```
#test image with id
##it is recommended to use different test images which were not used for training
test_data_location = image_folder

img = cv.imread(test_data_location+'2093.jpg',cv.IMREAD_GRAYSCALE) #load at gray
image = cv.resize(img, (60, 80),interpolation =cv.INTER_LINEAR)

ppcr = 8
ppcc = 8
hog_images_test = []
hog_features_test = []

blur = cv.GaussianBlur(image,(5,5),0)
fd_test,hog_img = hog(blur, orientations=8, pixels_per_cell=(ppcr,ppcc),cells_per
hog_images_test.append(hog_img)
hog_features_test.append(fd)

hog_features_test = np.array(hog_features_test)
y_pred_user = classifier.predict(scaler.transform(hog_features_test))
#print(plt.imshow(hog_images_test))
print(y_pred_user)
print("Predicted MaterCategory: ", mapper[mapper['class']==int(y_pred_user)]['mas
```

```
[0]
Predicted MaterCategory:  0    Apparel
Name: masterCategory, dtype: object
```

In [30]:
```
#test image HOG
plt.imshow(hog_img)
plt.show()
```



In [31]:
```
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import NearestNeighbors

scaler_global = MinMaxScaler()
final_features_scaled = scaler_global.fit_transform(hog_features)

neighbors = NearestNeighbors(n_neighbors=20, algorithm='brute')
neighbors.fit(final_features_scaled)

distance, potential = neighbors.kneighbors(scaler_global.transform(hog_features_t
print("Potential Neighbors Found!")
neighbors = []
for i in potential[0]:
    neighbors.append(i)

recommendation_list = list(df_labels.iloc[neighbors]['id'])
```

```
Potential Neighbors Found!
```

In [32]:
```
recommendation_list[0]
```

Out[32]:
```
2923
```

In [33]:
```
for i in pot
img = cv.imread(image_folder+str(7572)+'.jpg')
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
img.shape
plt.imshow(img)
```

```
  File "<ipython-input-33-30ef7d77a42f>", line 1
    for i in pot
                ^
SyntaxError: invalid syntax
```

In [ ]:
```
for i in range(6):
    img = cv.imread(image_folder+str(recommendation_list[i])+'.jpg')
    img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
    img.shape
    plt.imshow(img)
```

In [ ]:

In [ ]:

In [ ]: