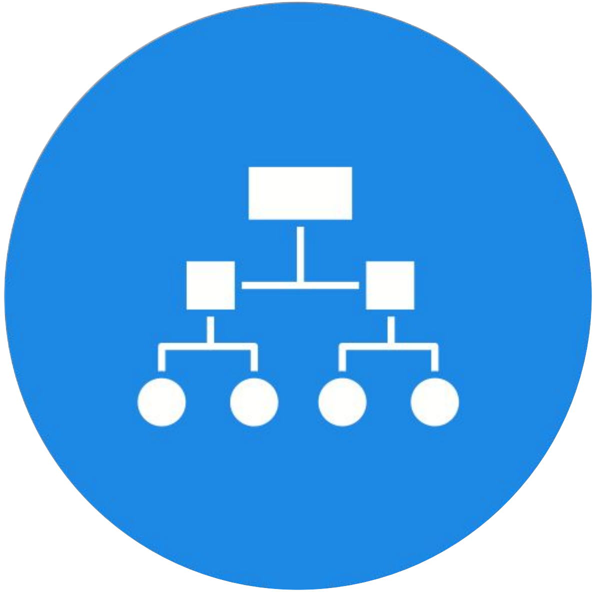


Fashion Feature Extraction Model

- Arjen GÖKDEMİR
- Azad AVŞAR
- Mehmet Umut ASLAN

LET’S START

Content

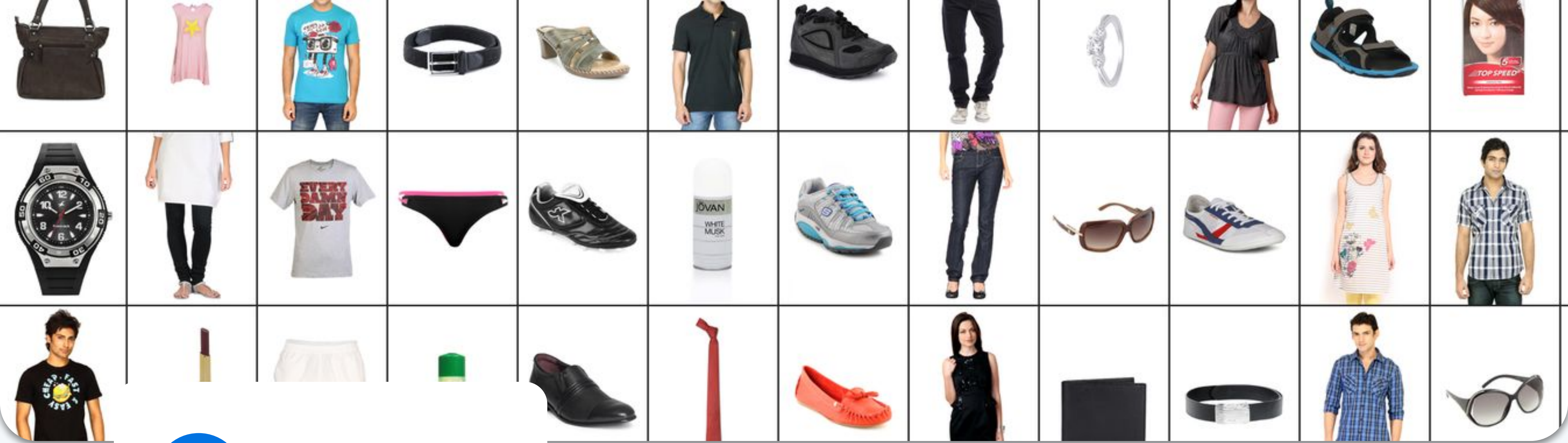
Fashion Feature Extraction Model (Fashion Recommendation Model)

Our Purpose

Our aim was to design a learning model that could learn and make predictions about the features of images. In this learning model, we chose two columns as targets, that is, our model will predict two features of that image based on one image.

For example, if we select the ''baseColour'' and ''articleType'' columns like we do. What we expect from our model is to respond "Blue T-Shirt" when faced with the visual of a Blue T-shirt.

Let's Examine the Dataset



Fashion Product Images (Small)

https://www.kaggle.com/datasets/paramaggarwal/fashion-product-images-small

- Images = 44.4k
- Styles = A csv file that contains the labels of images.

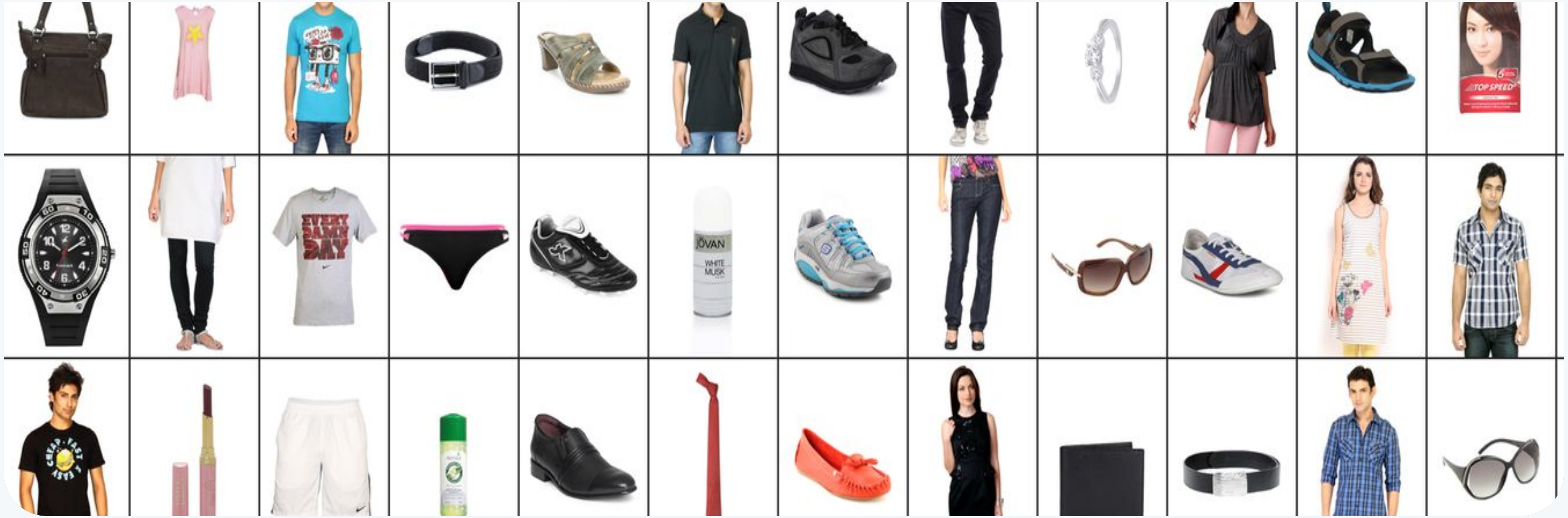
Dataset

Let's Look at the Details

	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	productDisplayName
0	15970	Men	Apparel	Topwear	Shirts	Navy Blue	Fall	2011.0	Casual	Turtle Check Men Navy Blue Shirt
1	39386	Men	Apparel	Bottomwear	Jeans	Blue	Summer	2012.0	Casual	Peter England Men Party Blue Jeans
2	59263	Women	Accessories	Watches	Watches	Silver	Winter	2016.0	Casual	Titan Women Silver Watch
3	21379	Men	Apparel	Bottomwear	Track Pants	Black	Fall	2011.0	Casual	Manchester United Men Solid Black Track Pants
4	53759	Men	Apparel	Topwear	Tshirts	Grey	Summer	2012.0	Casual	Puma Men Grey T-shirt

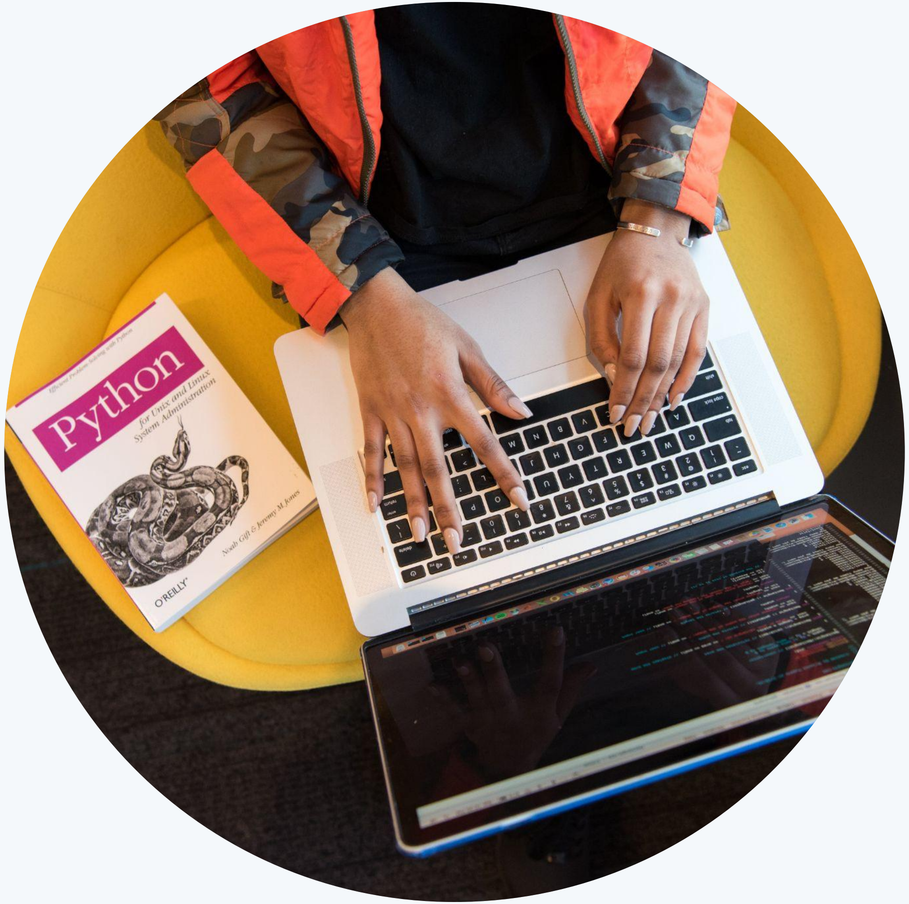
styles.csv

Images



What we Use?

Let's Examine Together



Libraries

NumPy

pandas

matplotlib

scikit learn

TensorFlow

OpenCV

Coding

Let's Examine the Codes

Data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('/kaggle/input/fashion-product-images-small/styles.csv',on_bad_lines = "skip")

df.head()
```

	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	productDisplayName
0	15970	Men	Apparel	Topwear	Shirts	Navy Blue	Fall	2011.0	Casual	Turtle Check Men Navy Blue Shirt
1	39386	Men	Apparel	Bottomwear	Jeans	Blue	Summer	2012.0	Casual	Peter England Men Party Blue Jeans
2	59263	Women	Accessories	Watches	Watches	Silver	Winter	2016.0	Casual	Titan Women Silver Watch
3	21379	Men	Apparel	Bottomwear	Track Pants	Black	Fall	2011.0	Casual	Manchester United Men Solid Black Track Pants
4	53759	Men	Apparel	Topwear	Tshirts	Grey	Summer	2012.0	Casual	Puma Men Grey T-shirt

```
df = df.dropna()
df.nunique()
df.columns
```

```
Index(['id', 'gender', 'masterCategory', 'subCategory', 'articleType',
       'baseColour', 'season', 'year', 'usage', 'productDisplayName'],
      dtype='object')
```

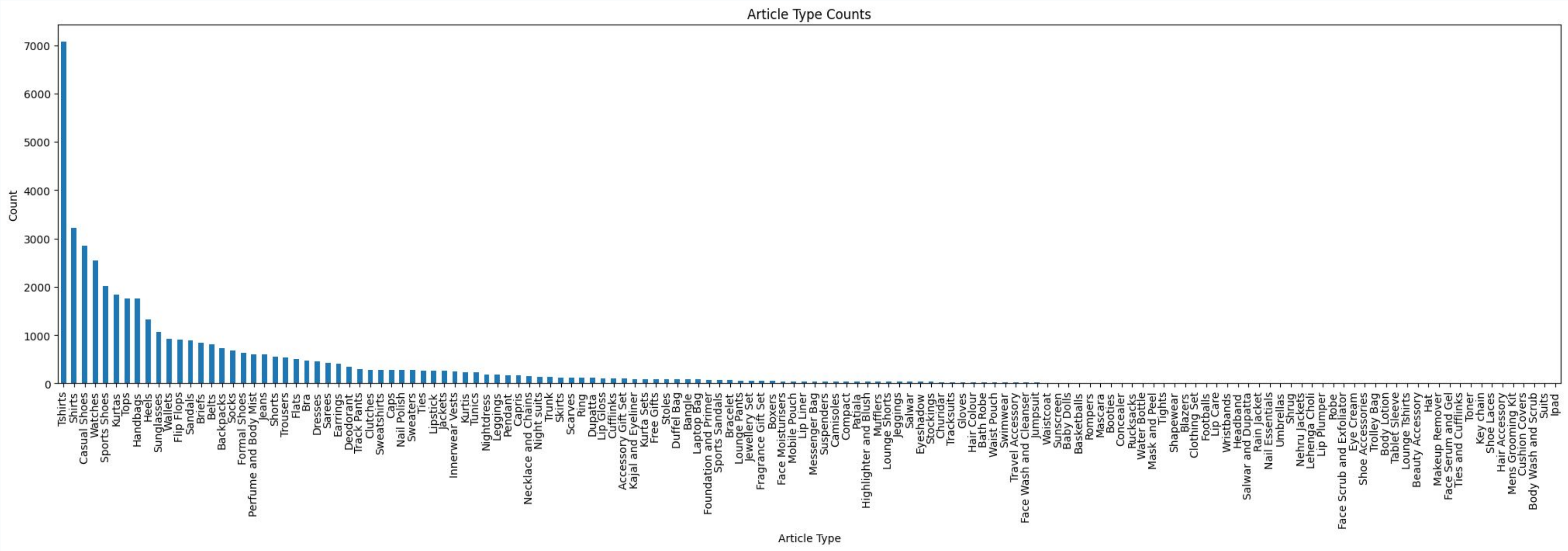
```
len(df)
```

```
44077
```

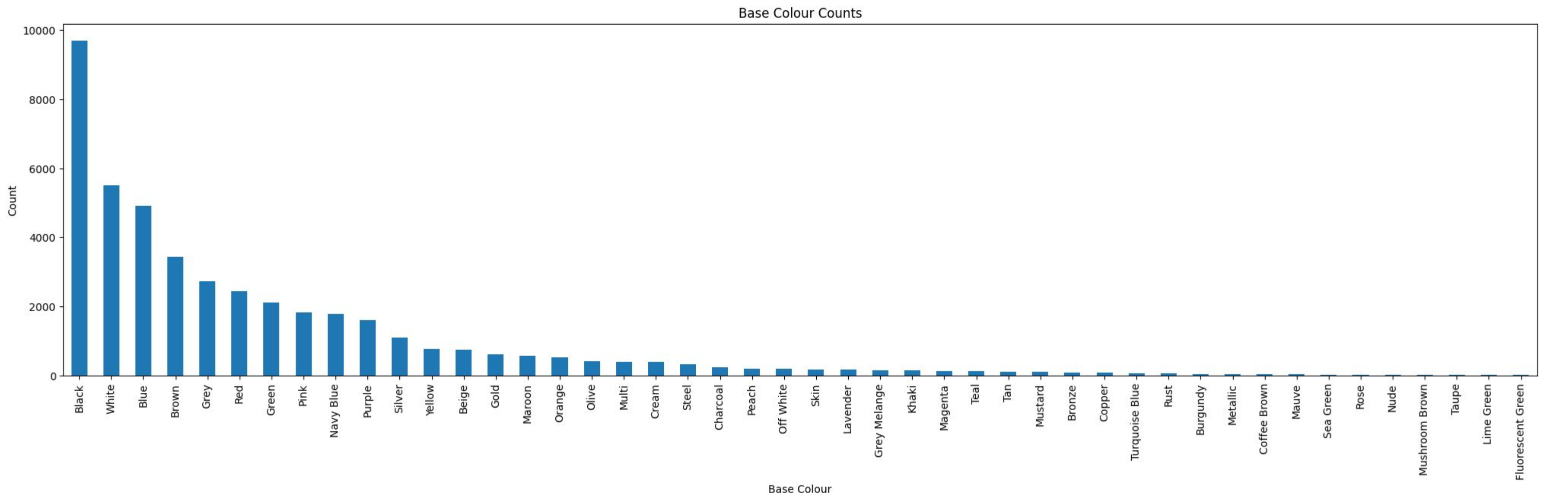
The columns that we chose

	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	productDisplayName
0	15970	Men	Apparel	Topwear	Shirts	Navy Blue	Fall	2011.0	Casual	Turtle Check Men Navy Blue Shirt
1	39386	Men	Apparel	Bottomwear	Jeans	Blue	Summer	2012.0	Casual	Peter England Men Party Blue Jeans
2	59263	Women	Accessories	Watches	Watches	Silver	Winter	2016.0	Casual	Titan Women Silver Watch
3	21379	Men	Apparel	Bottomwear	Track Pants	Black	Fall	2011.0	Casual	Manchester United Men Solid Black Track Pants
4	53759	Men	Apparel	Topwear	Tshirts	Grey	Summer	2012.0	Casual	Puma Men Grey T-shirt

Article Type



Base Color



Base Colour Counts
Count
0
2000
4000
6000
8000
10000
Black
White
Blue
Brown
Grey
Red
Green
Pink
Navy Blue
Purple
Silver
Yellow
Beige
Gold
Maroon
Orange
Olive
Multi
Cream
Steel
Charcoal
Peach
Off White
Skin
Lavender
Grey Melange
Khaki
Magenta
Teal
Tan
Mustard
Bronze
Copper
Turquoise Blue
Rust
Burgundy
Metallic
Coffee Brown
Mauve
Sea Green
Rose
Nude
Mushroom Brown
Taupe
Lime Green
Fluorescent Green
Base Colour

Data Filtering

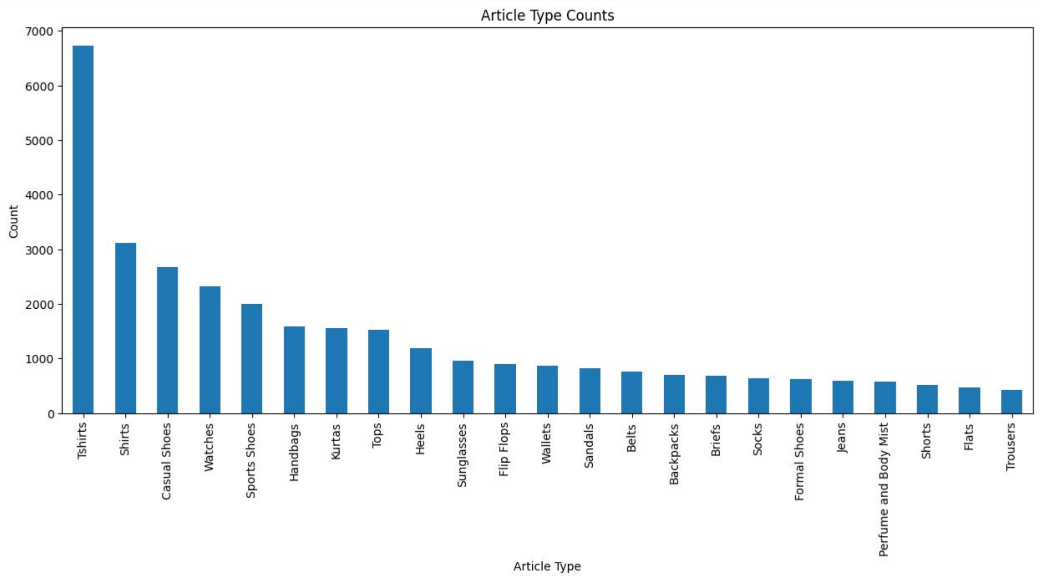
```
value_counts = df['articleType'].value_counts()

indexes = value_counts.index

values = value_counts.values

for i in range(len(value_counts)):

    if values[i] <500:
        break

types_used = indexes[:i]

print('Article types used: ',types_used)
```

```
Article types used:  Index(['Tshirts', 'Shirts', 'Casual Shoes', 'Watches', 'Sports Shoes',
       'Kurtas', 'Tops', 'Handbags', 'Heels', 'Sunglasses', 'Wallets',
       'Flip Flops', 'Sandals', 'Briefs', 'Belts', 'Backpacks', 'Socks',
       'Formal Shoes', 'Perfume and Body Mist', 'Jeans', 'Shorts', 'Trousers',
       'Flats'],
      dtype='object', name='articleType')
```

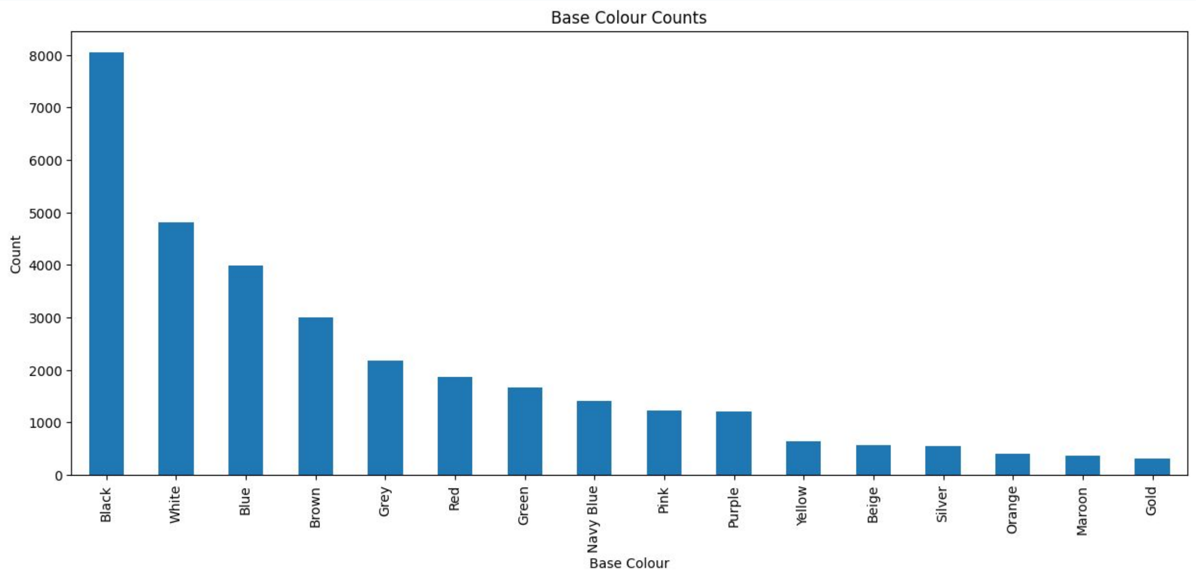
```
value_counts = df['baseColour'].value_counts()

indexes = value_counts.index

values = value_counts.values

for i in range(len(value_counts)):

    if values[i] <500:
        break

colours_used = indexes[:i]
print('Base Colours used: ',colours_used)
```

```
Base Colours used:  Index(['Black', 'White', 'Blue', 'Brown', 'Grey', 'Red', 'Green', 'Pink',
       'Navy Blue', 'Purple', 'Silver', 'Yellow', 'Beige', 'Gold', 'Maroon',
       'Orange'],
      dtype='object', name='baseColour')
```

Filtered Article Type



Filtered Base Color



Preparing the Images List

- Create a list named ‘‘data’’
- Set the initial images shape to the 80,60
- IX = 80 , IY = 60
- Reshape the images with respect to IX and IY. If encounter any id’s that doesn’t exist, print their id’s.
- Convert images to array and add them to the list
- The funcs. that we use: ‘‘ cv2.imread’’ , ‘‘ cv2.resize’’ , ‘‘ img_to_array’’

Let's Continue with Normalization →

Normalization

This normalization converts pixel values between 0 and 255 to between 0 and 1..In this way, the learning process will be more consistent. Also we convert the labels list to the np.array list.

```
import numpy as np


data = np.array(data, dtype="float") / 255.0
labels = np.array(labels)

print(labels)
```

```
[['Shirts' 'Navy Blue']
 ['Jeans' 'Blue']
 ['Watches' 'Silver']
 ...
 ['Tshirts' 'Blue']
 ['Perfume and Body Mist' 'Blue']
 ['Watches' 'Pink']]
```

Multi Label Binarizer

MultiLabelBinarizer (MLB) is a preprocessing tool used especially for multilabel classification problems.
MultiLabelBinarizer places each label in a separate column and converts these columns to binary format.

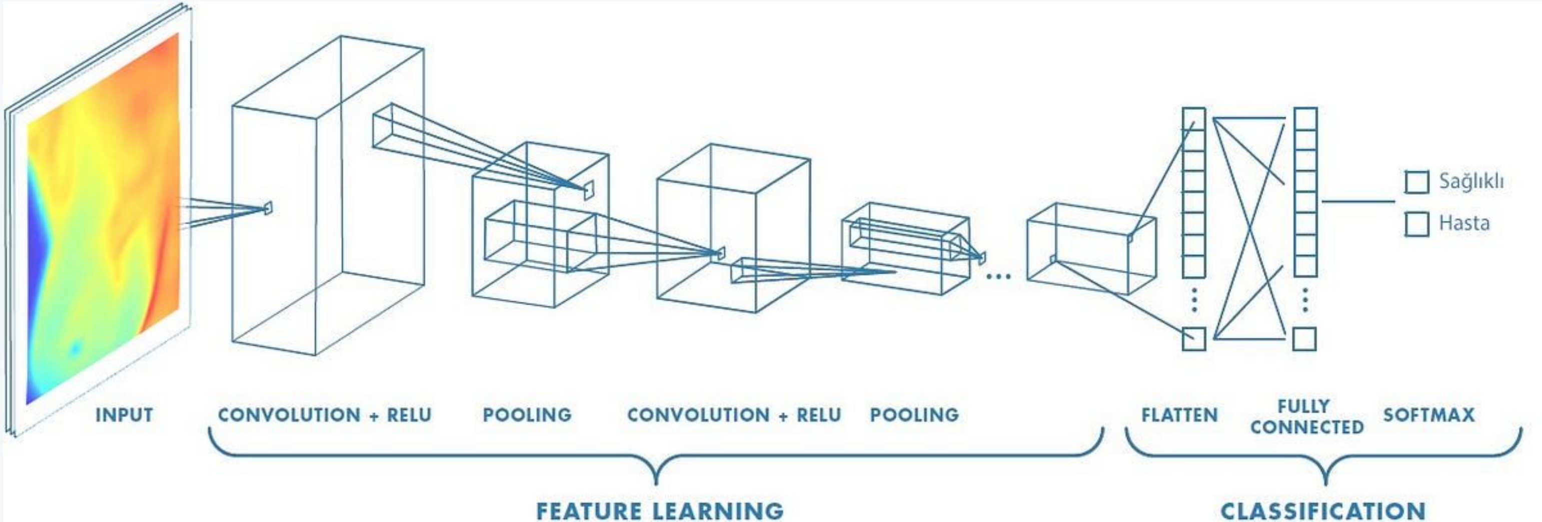
```
from sklearn.preprocessing import MultiLabelBinarizer


mlb = MultiLabelBinarizer()
labels = mlb.fit_transform(labels)

print(mlb.classes_)
print(labels[0])
```

```
['Backpacks' 'Beige' 'Belts' 'Black' 'Blue' 'Briefs' 'Brown'
 'Casual Shoes' 'Flats' 'Flip Flops' 'Formal Shoes' 'Gold' 'Green' 'Grey'
 'Handbags' 'Heels' 'Jeans' 'Kurtas' 'Maroon' 'Navy Blue' 'Orange'
 'Perfume and Body Mist' 'Pink' 'Purple' 'Red' 'Sandals' 'Shirts' 'Shorts'
 'Silver' 'Socks' 'Sports Shoes' 'Sunglasses' 'Tops' 'Trousers' 'Tshirts'
 'Wallets' 'Watches' 'White' 'Yellow']
[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 1 0 0 0 0 0 0 0 0 0 0
 0 0]
```

The two 1 values are the equivalent of the first two elements in our label list.
We can check this from the class list we call with mlb.classes.
One indexes = 20,27 / Labels[0] = Shirts , Navy Blue

CNN

Since we are working on images, we need to design a CNN model before moving on to the learning phase.



What Does the Structure of CNN Consist of?

Convolutional Layer — Used to detect features

```
model.add(Conv2D(32, (3, 3), padding="same", input_shape=inputShape))
model.add(Activation("relu"))
```

Pooling Layer — Reduces the number of weights and checks compliance

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

Flattening Layer — Prepares data for Classic Neural Network

```
model.add(Flatten())
```

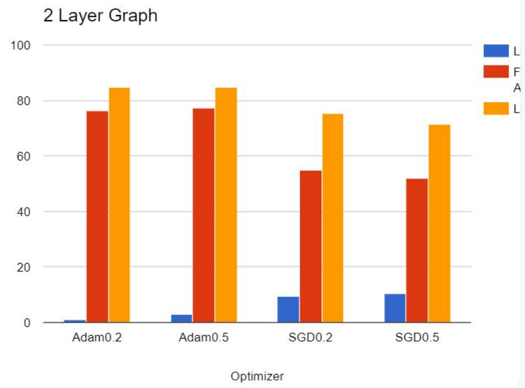
Fully-Connected Layer — Standard Neural Network used in classification

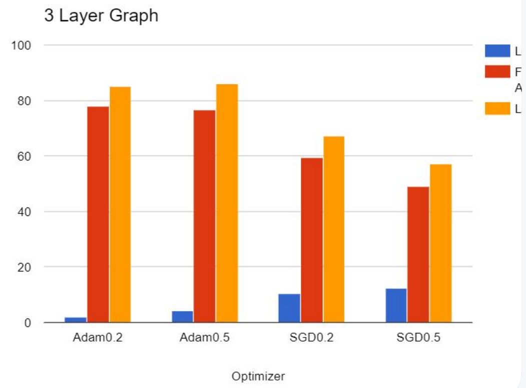
```
model.add(Dense(128))
model.add(Activation('relu'))

out = len(mlb.classes_)
model.add(Dense(out))
model.add(Activation('sigmoid'))
```

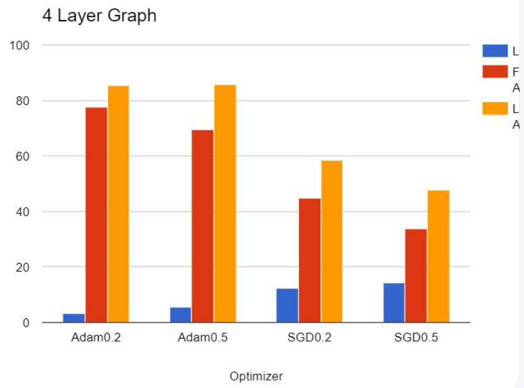
Determining the CNN Model We Will Use

- Here are the steps we took to determine the model we will use.
- Designing models with different convolution layers.(2,3,4)
- Testing different variations on these models by playing with dropout and optimizer.
- Finally, finding the most successful model, dropout and optimizer depending on the accuracy value.

Models We Created







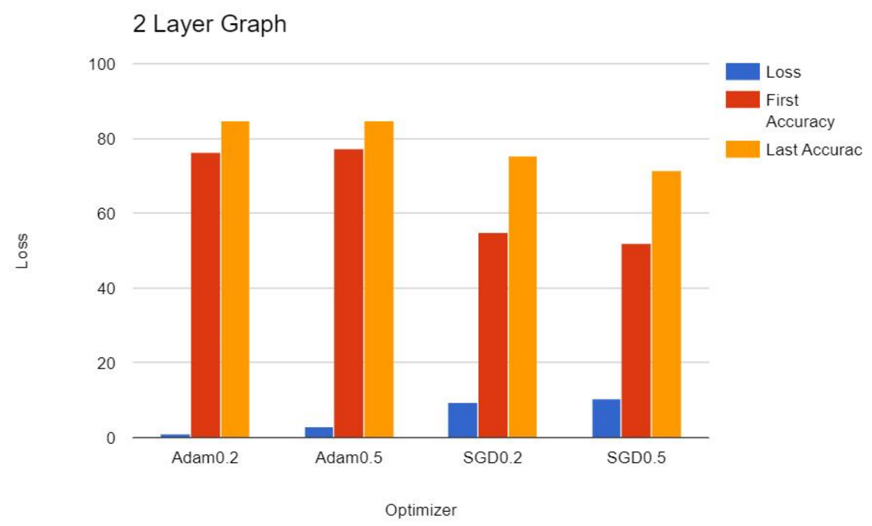
2 Layer Model →

3 Layer Model →

4 Layer Model →

Combinations

- Adam + 0.2 Dropout
- Adam + 0.5 Dropout
- SGD + 0.2 Dropout
- SGD + 0.5 Dropout

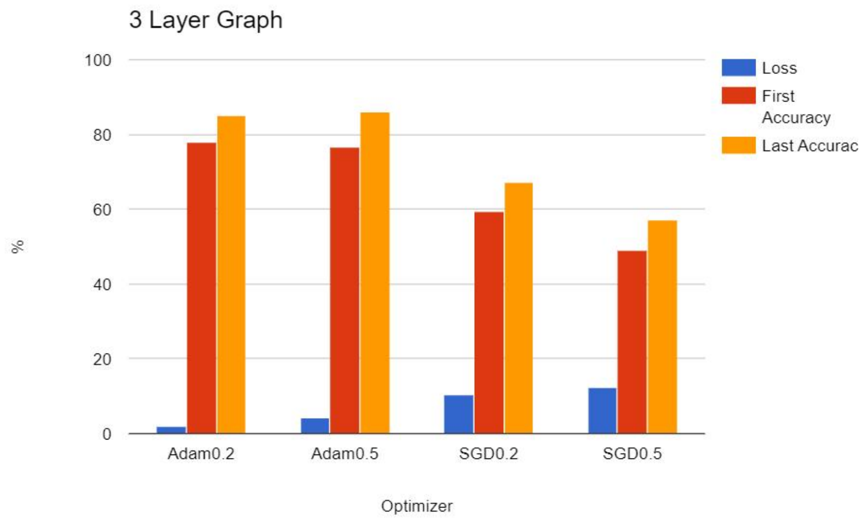


2 Layer Model

3 Layer Model →

Combinations

- Adam + 0.2 Dropout
- Adam + 0.5 Dropout
- SGD + 0.2 Dropout
- SGD + 0.5 Dropout

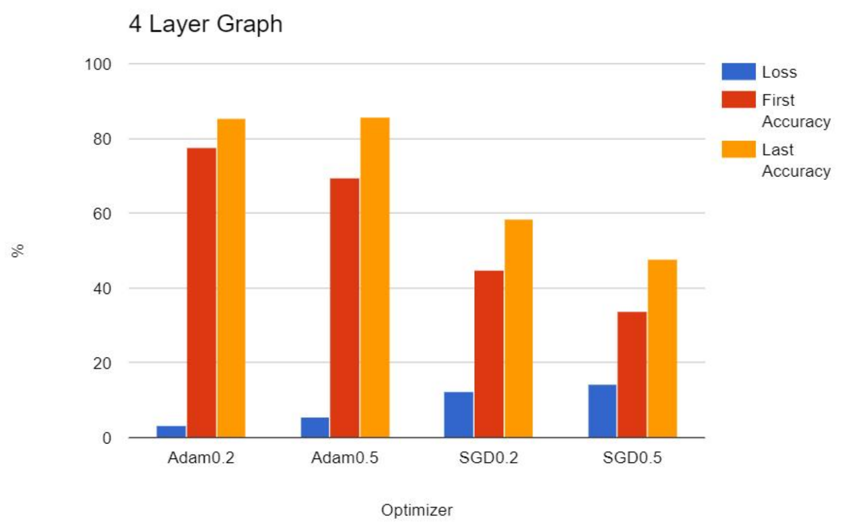


3 Layer Model

4 Layer Model

Combinations

- Adam + 0.2 Dropout
- Adam + 0.5 Dropout
- SGD + 0.2 Dropout
- SGD + 0.5 Dropout

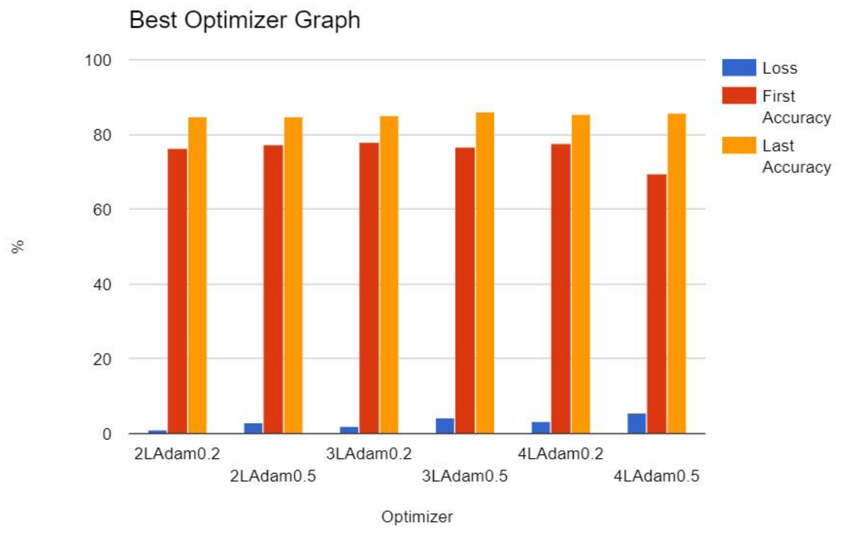


4 Layer Model

Comparing Models

Result

The Highest Accuracy Value was Obtained in the 3–Layer Model Created Using Adam Optimizer and 0.5 Dropout.



Comparing Models

4 Layer Model →

Our Best CNN Model

```python
from tensorflow.keras.layers import Flatten, Dropout, Dense, Activation, Conv2D, MaxPooling2D
from tensorflow.keras.models import Sequential


inputShape = (IY, IX, 3)

model = Sequential()

model.add(Conv2D(32, (3, 3), padding="same", input_shape=inputShape))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))


model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))


model.add(Conv2D(128, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))
model.summary()

model.add(Flatten())

model.add(Dense(128))
model.add(Activation('relu'))

out = len(mlb.classes_)
model.add(Dense(out))
model.add(Activation('sigmoid'))


model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['mse'])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 60, 80, 32)        896

 activation (Activation)     (None, 60, 80, 32)        0

 max_pooling2d (MaxPooling2  (None, 30, 40, 32)        0
 D)

 dropout (Dropout)           (None, 30, 40, 32)        0

 conv2d_1 (Conv2D)           (None, 28, 38, 64)        18496

 activation_1 (Activation)   (None, 28, 38, 64)        0

 max_pooling2d_1 (MaxPoolin  (None, 14, 19, 64)        0
 g2D)

 dropout_1 (Dropout)         (None, 14, 19, 64)        0

 conv2d_2 (Conv2D)           (None, 12, 17, 128)       73856

 activation_2 (Activation)   (None, 12, 17, 128)       0

 max_pooling2d_2 (MaxPoolin  (None, 6, 8, 128)         0
 g2D)

 dropout_2 (Dropout)         (None, 6, 8, 128)         0

=================================================================
Total params: 93248 (364.25 KB)
Trainable params: 93248 (364.25 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Testing

Let's Test the Model

Split Data

```
(trainX, testX, trainY, testY) = train_test_split(data,labels, test_size=0.2, random_state=42)
```

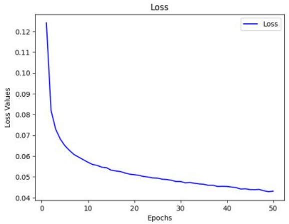
- We defined test size as 0.2
- Length of test = 6449
- Length of train = 25792

Get Results

Learning process:

(Batch size = 32 / Epoch = 50)

Our final loss: 0.0431



Let's Test the Model

Prediction of the Model

Models Prediction After Training

```
True labels:  ('Black', 'Sports Shoes')  Predicted labels:  ('Sports Shoes',)
True labels:  ('Kurtas', 'Orange')  Predicted labels:  ('Kurtas', 'Maroon', 'Red')
True labels:  ('Blue', 'Perfume and Body Mist')  Predicted labels:  ('Blue', 'Perfume and Body Mist')
True labels:  ('Black', 'Watches')  Predicted labels:  ('Black', 'Watches')
True labels:  ('Black', 'Wallets')  Predicted labels:  ('Black', 'Wallets')
True labels:  ('Black', 'Trousers')  Predicted labels:  ('Black', 'Navy Blue', 'Trousers')
True labels:  ('Black', 'Sunglasses')  Predicted labels:  ('Black', 'Sunglasses')
True labels:  ('Maroon', 'Shirts')  Predicted labels:  ('Maroon', 'Shirts')
True labels:  ('Briefs', 'Maroon')  Predicted labels:  ('Briefs',)
True labels:  ('Black', 'Watches')  Predicted labels:  ('Black', 'Watches')
```

Results

```
202/202 [==============================] - 1s 3ms/step
correct:  9923
missing/wrong:  2975
Accuracy:  0.7693440843541635
```

Detection of Errors

- After viewing the initial prediction process of the model, we noticed that in some predictions, the positions of the columns shifted both in the correct part and in the prediction part.
- We also saw that some predictions included one or three predictions instead of two.
- For this reason, we made the model's predictions visually beautiful by applying a filter.

```
True labels:  ('Black', 'Sports Shoes')  Predicted labels:  ('Sports Shoes',)
True labels:  ('Kurtas', 'Orange')  Predicted labels:  ('Kurtas', 'Maroon', 'Red')
True labels:  ('Blue', 'Perfume and Body Mist')  Predicted labels:  ('Blue', 'Perfume and Body Mist')
True labels:  ('Black', 'Watches')  Predicted labels:  ('Black', 'Watches')
True labels:  ('Black', 'Wallets')  Predicted labels:  ('Black', 'Wallets')
True labels:  ('Black', 'Trousers')  Predicted labels:  ('Black', 'Navy Blue', 'Trousers')
True labels:  ('Black', 'Sunglasses')  Predicted labels:  ('Black', 'Sunglasses')
True labels:  ('Maroon', 'Shirts')  Predicted labels:  ('Maroon', 'Shirts')
True labels:  ('Briefs', 'Maroon')  Predicted labels:  ('Briefs',)
True labels:  ('Black', 'Watches')  Predicted labels:  ('Black', 'Watches')
```

Filtering Process

Detection, Filtration and Revision of Incorrect Pairs

Let's Filter

Filtering Process-1

Deletion of Estimates with Missing Values::

- Prediction Sets were Checked One by One
- Prediction Tags Containing Less Than 2 Elements Were Detected
- Detected Tags Deleted from Prediction and Test Sets

Filtering Process-2

Correcting the Order of Incorrectly Sorted Predictions:

- Prediction and Test Sets were Checked One by One
- Labels Without Color In The First Row Were Detected
- Detected Labels were Rewritten in the Correct Order in the Prediction and Test Sets

Let's Test the Filtered Model

Results After Filtering

Models Prediction After Filtering

```
True labels:  ('Blue', 'Perfume and Body Mist')  Predicted labels:  ('Blue', 'Perfume and Body Mist')
True labels:  ('Black', 'Watches')  Predicted labels:  ('Black', 'Watches')
True labels:  ('Black', 'Wallets')  Predicted labels:  ('Black', 'Wallets')
True labels:  ('Black', 'Sunglasses')  Predicted labels:  ('Black', 'Sunglasses')
True labels:  ('Maroon', 'Shirts')  Predicted labels:  ('Maroon', 'Shirts')
True labels:  ('Black', 'Watches')  Predicted labels:  ('Black', 'Watches')
True labels:  ('Brown', 'Wallets')  Predicted labels:  ('Brown', 'Wallets')
True labels:  ('White', 'Tops')  Predicted labels:  ('White', 'Tops')
True labels:  ('Red', 'Tshirts')  Predicted labels:  ('Red', 'Tshirts')
True labels:  ('Brown', 'Belts')  Predicted labels:  ('Brown', 'Belts')
```

Results

```
correct:  8258
missing/wrong:  1402
Accuracy:  0.8548654244306418
```

Fashion Feature Extraction Model

Thank You

Check Out
Our Blog

mlp.lptn.name.tr