Athens University of Economics and Business

School of Business

Department of Management Science & Technology

Master of Science in Business Analytics

Program:	Full-time
Quarter:	3rd (Spring Quarter)
Course:	Machine Learning & Content Analytics
Assignment:	Image Classification for Outfit Suggestions
Students (Registration №):	Vlassis Georgios–Konstantinos (f2822203), Sakellaris Emmanouil (f2822215), Souflas Eleftherios-Efthymios (f2822217), Tsapatsaris Dimitrios (f2822218)

Table of Contents

Image Classification for Outfit Suggestions

Introduction

In the dynamic landscape of machine learning and data-driven solutions, the ability to effectively classify and analyze data has become increasingly vital. This report presents a comprehensive exploration of our project, which centers around the classification of fashion products using state-of-the-art machine learning techniques.

Our project

DressMeUp is an Artificial Intelligence (AI) application that creates outfit combinations and recommendations for purchases. It uses existing pieces of clothing (articles) in order to create outfits based on a certain occasion, time of day or weather conditions. It scouts the web for existing and upcoming fashion trends and learns what clothing combinations work together. As an API, it can be integrated into the websites of clothing e-shops and match selected pieces of clothing from the e-shops to the ones from the user's wardrobe. Through the application, users can also sell their clothes and buy from other users or from partnered e-shops.

Our Vision/Goals

Our vision is, through our application, to address the need for full exploitation of the user's wardrobe, creating outfits automatically from registered to the application clothes. The services proposed by our business idea are:

- Spontaneous matching of the already registered to the application clothes.
- Creation of a weekly plan of the user's dressing obligations (e.g., job, meeting, vacations, dining) based on the registered clothes.
- Selling individual's clothes.
- Recommendations for purchase of new clothes by online stores or other users' clothes based on unmatched, poorly matched or selected by the user clothes.

The innovation provided by our business is two-folded:

- Firstly, with its AI algorithm it will provide recommendations taking into consideration the latest fashion trends, the local weather of the user, the formality and time of day of each occasion.
- Secondly, its Application Programming Interface (API) will be embedded to any clothing e-shop website that wish to provide to its customers the functionality of matching a chosen e-shop cloth to the wardrobe clothes of its customer.

There already exist in the market direct competition of businesses like "Acloset", "OpenWardrobe" and "Whering". However, our proposed idea is different from our competitors in the aspects below:

Functionality	**DressMeUp**	**Acloset**	**Whering**	**OpenWardrobe**
Add clothes	✓	✓	✓	✓
Make outfits	✓	✓	✓	✓
Plan outfits	✓	✓	✓	✓
Sell clothes	✓	✓		
Buy clothes	✓	✓	✓	
Track clothes usage	✓			

Functionality	DressMeUp	Acloset	Whering	OpenWardrobe
AI uses weather forecast	✓		✓	
AI uses occasion formality & time of day	✓			
E-Shops embedded API	✓			

Methodology

Our methodology is centered around utilizing data and machine/deep learning techniques to categorize and analyze clothing items. The goal is to enhance the understanding and classification of fashion products based on their attributes, starting by preprocessing and structuring the data, focusing on specific clothing categories e.g., usage, and seasonal attributes. Then, machine learning and deep learning models are employed to classify and extract meaningful insights from the fashion dataset. The model with the best evaluation out of these models is then utilized for making the classifications from users' clothes input images to our application. Based on these classifications, along with any updates that the user wishes to make to the proposed by the model values, our application will provide all forementioned services to its users.

Data Collection

The data source that our data mostly came from was Fashion Product Images Dataset from Kaggle, which contains 44 thousands products with multiple category labels, descriptions and high-resolution images. Also, from thomascamminady GitHub repository, we obtained the digital version of 'A Dictionary of Color Combinations' by Sanzo Wada book's collection of color combinations, in order to utilize for combining properly the classified clothes.

Dataset Overview

The dataset contains 44,446 products. Each one of these products is identified by an ID. A map to all products exists in the 'styles' CSV file, containing various features for each product. The features that we found most useful for our project and used for classifying clothes from input images were 'gender', 'subCategory', 'articleType', 'season', and 'usage'. The images of 44,441 of these products can be fetched from 'images/{ID}.jpg'. 'Gender' attribute has 5 unique values, 'subCategory' has 45, 'articleType' has 143, 'season' has 5, and 'usage' has 9 unique values. The images are of resolution 1800x2400. Also, in the color combination repository exist 348 combinations (in the 'combinations' JSON file) from 157 different colors in the 'colors' JSON file.

Data Processing

Firstly, because we will only match top and bottom-wear clothes (e.g., trousers and T-shirts), we deleted all other categories, like watches, socks and shoes from the subCategory column, except the 'Loungewear and Nightwear' subCategory from which we kept the 'Shorts' articleType, renaming its 'subCategory' column value to 'Bottomwear'. Also, from the usage column, we discarded ethnic clothes and kept only casual, formal, sports, smart casual, party, and travel clothes. Gender column included unisex, boys' and girls' clothes, which we did not use for our model (we only used men and women clothes). We also created a new column holding the image path, discarding products that no image was included in the image folder.

Then, we imputed null values found in usage and season columns of the remaining rows of the dataset. The imputation schema that we followed was rather simple, since we do not have numerical values. Null values in usage column were found in three shirt products and so we updated them with the ‘Formal’ category value. In season column, a null value was found in a T-shirt product and so we updated it with the ‘Summer’ category value.

The distribution of clothes per feature included, prior to the application of data augmentation can be found on the figures below (Figure 1, Figure 2, Figure 3, Figure 4, and Figure 5).

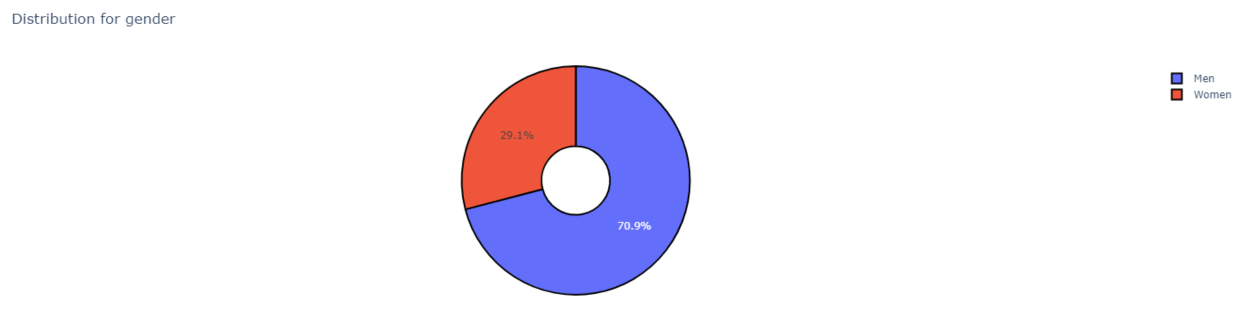


Figure 1 - Distribution for gender prior to data augmentation

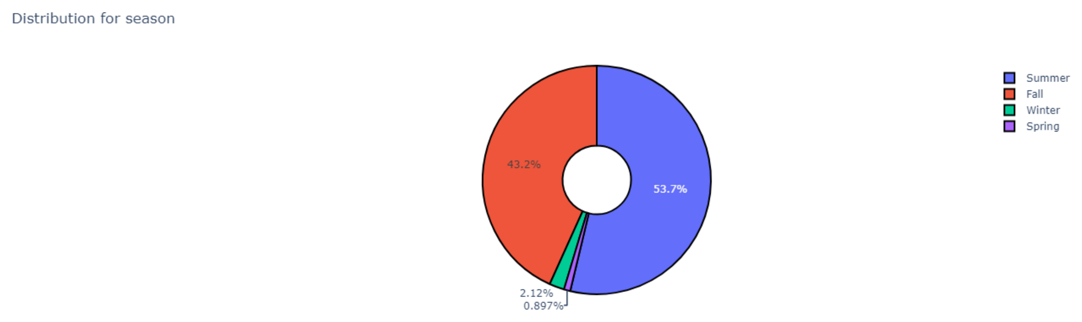


Figure 2 - Distribution for season prior to data augmentation

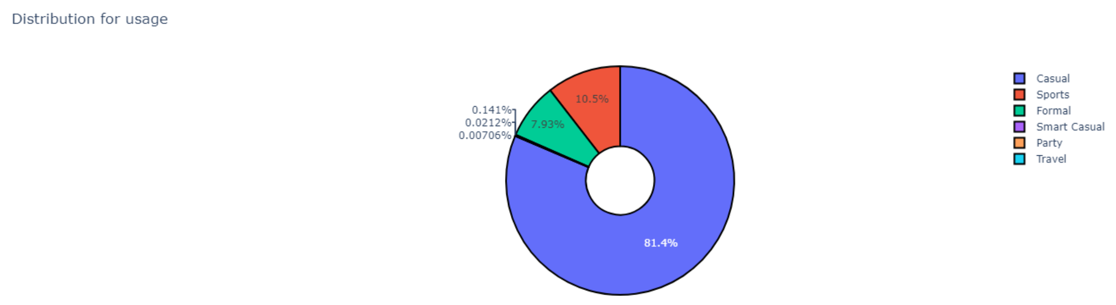


Figure 3 - Distribution for usage prior to data augmentation

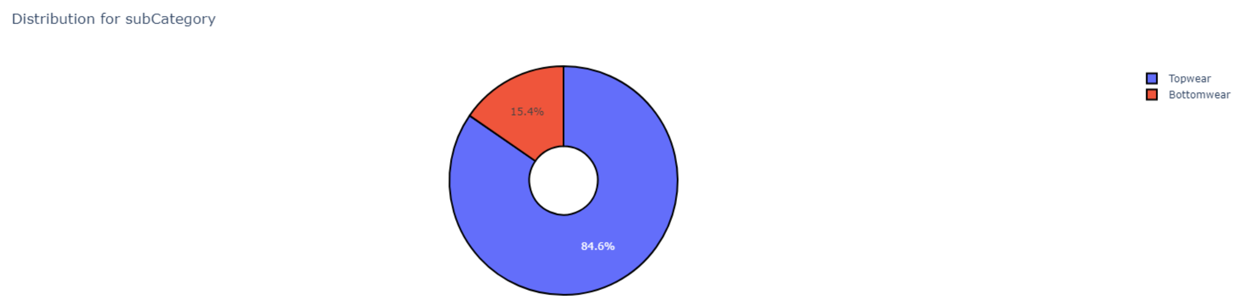


Figure 4 - Distribution for subCategory prior to data augmentation

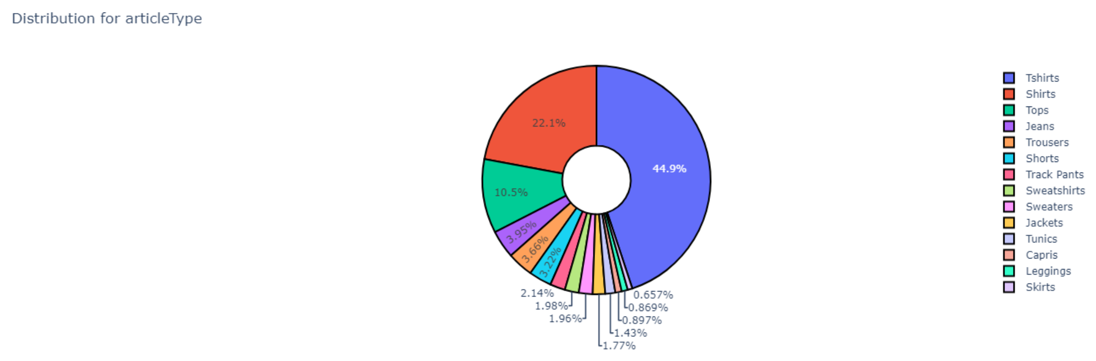


Figure 5 - Distribution for articleType prior to data augmentation

After that, because 'Smart Casual', 'Party', and 'Travel' clothes were actually casual clothes and had only a few samples, we updated their usage value to 'Casual'. Similarly, because 'Winter' and 'Spring' clothes had only a few samples and because Spring/Summer (SS) and Autumn/Winter (AW) are the two larger and main runway seasons presented at fashion weeks according to Retail Dogma (Mahmoud, 2023), we updated the respective values to 'Spring/Summer' and 'Fall/Winter'.

Then, because there existed a discrepancy in the number of clothes per articleType (e.g., T-shirts, Jeans, etc.) included in the dataset, ranging from a hundred to 6k, we applied image rotations, scaling, and horizontal flips to increase the number of images to 1k per articleType. Thus, we ended up with a 14k dataset (14 articleType categories of 1k each). The distribution of clothes per feature included, after the application of data augmentation can be found on the figures below (Figure 6, Figure 7, Figure 8, Figure 9, and Figure 10).

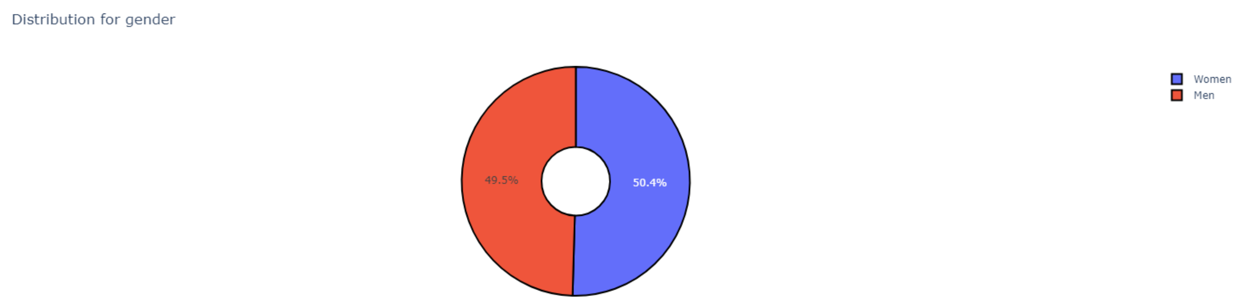


Figure 6 - Distribution for gender after data augmentation

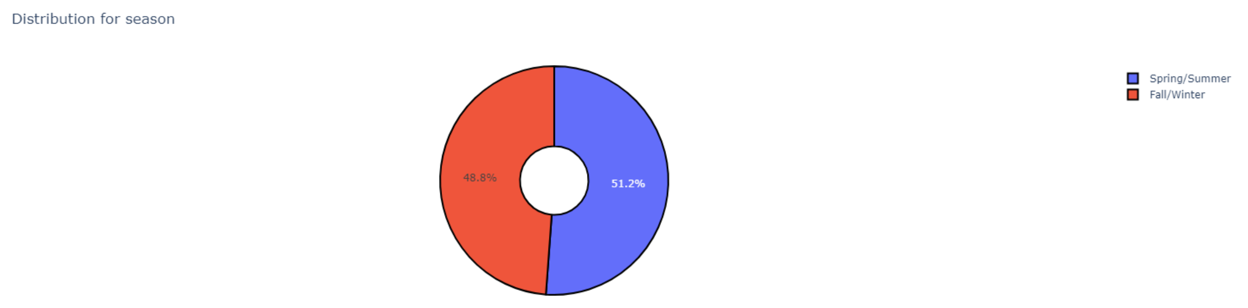


Figure 7 - Distribution for season after data augmentation

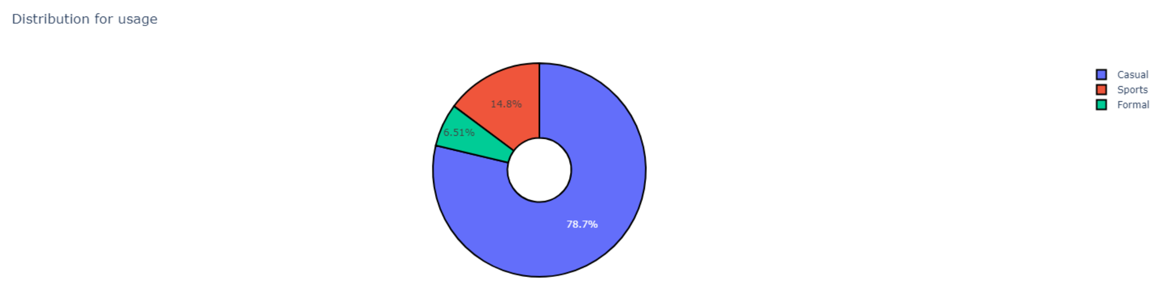


Figure 8 - Distribution for usage after data augmentation

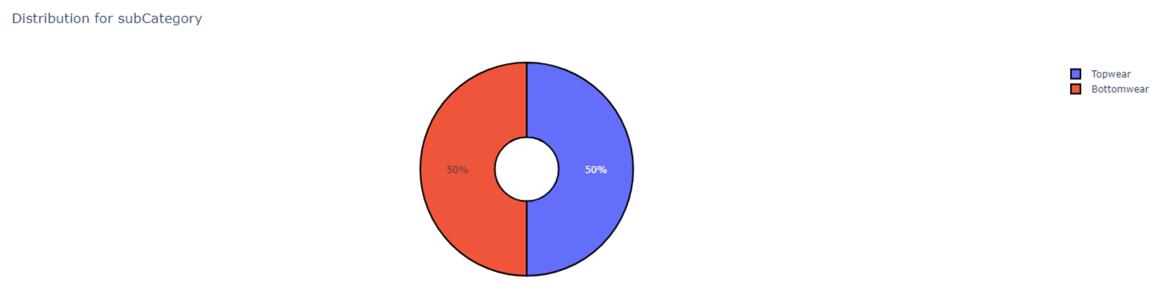


Figure 9 - Distribution for subCategory after data augmentation

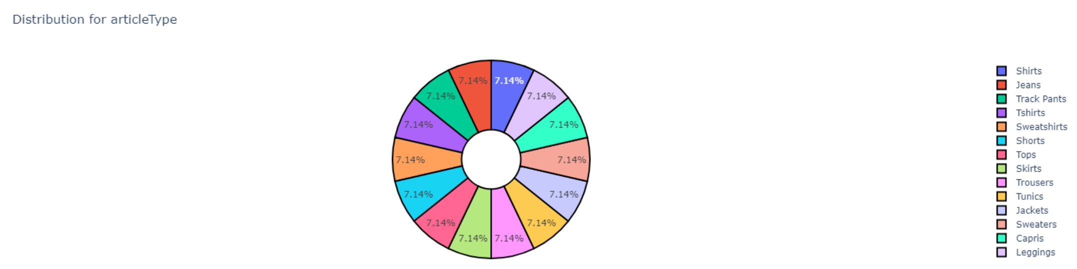


Figure 10 - Distribution for articleType after data augmentation

Finally, we encoded the labels of each category by alphabetic order using an integer value starting from 0 e.g., Casual, Formal, Sports to 0, 1, 2 and then stored the data frame to a csv file to be easily distributed across project members for parallel execution of the work needed to be done.

Algorithms

The algorithms employed in our project are:

1. Stochastic Gradient Descent (SGD) Classifier with hinge loss and squared Euclidean norm L2 penalty, effectively applying Support Vector Machine (SVM) classifier. This estimator implements regularized linear models with stochastic gradient descent (SGD) learning: the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate) (Scikit-Learn, 2023).
2. Passive Aggressive Classifier with hinge loss. It belongs to a family of algorithms for large-scale learning. It does not require a learning rate, but includes a regularization parameter C (Scikit-Learn, 2023).
3. Convolutional Neural Network (CNN) with transfer learning from ResNet50 architecture. In our project, we leverage the power of Convolutional Neural Networks (CNNs) for image classification tasks. Specifically, we utilize transfer learning from the ResNet50 architecture. ResNet stands for Residual Network and is a specific type of convolutional neural network (CNN) introduced in the 2015 paper "Deep Residual Learning for Image Recognition" (He, Zhang, Ren, & Sun, 2023). Residual neural networks are a type of artificial neural network (ANN) that form networks by stacking residual blocks. ResNet50 is a 50-layer convolutional neural network (48 convolutional layers, one MaxPool layer, and one average pool layer) (Datagen, 2023).
4. Convolutional Neural Network (CNN) with transfer learning from EfficientNetV2-S architecture. In addition to ResNet50, we employ another powerful Convolutional Neural Network architecture, EfficientNetV2S, for our image classification tasks. Efficient nets are the family of neural networks with the baseline model constructed with Neural Architecture Search. Neural Architecture Search is a technique for automating the design of artificial neural networks. The type of artificial neural network that can be designed depends on the search space (Hasty, 2023). It is a type of Convolutional Neural Network that has faster training speed and better parameter efficiency than previous models (Tan & Le, 2023). EfficientNetV2 models train much faster than state-of-the-art models while being up to 6.8× smaller (Tsang, 2023). EfficientNetV2S is the smaller model of the V2 models and is known for its efficiency in terms of computational resources while maintaining excellent performance. By utilizing transfer learning from EfficientNetV2S, we take advantage of its ability to capture intricate features from images, enhancing the accuracy and robustness of our fashion item classification model. This approach allows us to leverage state-of-the-art architecture without the need for extensive computational resources during training.
5. K-Means Clustering to cluster pixels in 5 clusters in a specified image region and extract the most dominant color (the color with the greater number of pixels) out of them.

Notes: Transfer learning involves using a pre-trained neural network, such as a ResNet50 or an EfficientNetV2S, which has been trained on a large dataset for various image recognition tasks. By fine-tuning this pre-trained network on our specific fashion dataset, we can harness the knowledge and features it has learned from a wide range of images, enabling our model to effectively classify fashion items based on their visual attributes.

Experiments – Setup, Configuration

For all models, we applied a load and preprocess function for images to load them, resize them to a common size (we used 224x224 pixels) and normalize pixel values to the range [0,1]. We also divided the dataset to training, validation, and test sets with a ratio of 80%, 10%, and 10% of the original dataset respectively, stratifying on an artificial column created from the five output/target columns. Stratification ensures that each subset contains a representative sample from each class, maintaining the same class distribution as the original dataset.

At start, we applied the first two models in order to have a baseline model that is less resource-hungry before applying CNNs, which are known to have better results than typical machine learning models with larger training cost (time and resources). For these models, no special hardware or software configuration needed, apart from applying the scikit-learn's 'partial_fit' method, which provides the ability to learn incrementally from a mini-batch of instances (sometimes called "online learning"), which is a good strategy to scale computationally bigger data. This method is key to out-of-core learning as it guarantees that at any given time there will be only a small number of instances in the main memory. Multi target classification was applied with the use of scikit-learn's MultiOutputClassifier. This strategy consists of fitting one classifier per target. This is a simple strategy for extending classifiers that do not natively support multi-target classification. MultiOutputClassifier was applied with the maximum available number of processes/threads to run in parallel.

For the first model, scikit-learn's SGDClassifier was used, as the model of the MultiOutputClassifier, with hinge loss, squared Euclidean norm L2 penalty, shuffling of data and all other hyperparameters in their default values, in order to effectively apply SVM. The training lasted approximately for 1.5 minute.

For the second model, scikit-learn's PassiveAggressiveClassifier was used with hinge loss, shuffling of data and all other hyperparameters in their default values. The training lasted for approximately 1.5 minute.

Then, we started training our CNN models. Because none of the members of this project owned a PC with an NVIDIA GPU installed, at first, we started training our models on our CPU, but the training time of a single epoch lasted from 50 minutes to 1 hour and 5 minutes, depending on the pre-trained CNN, layers, and the tuning of hyperparameters. Then, because we had internal Intel GPU, we tried the Direct ML Plugin for Tensorflow 2. We followed all instructions mentioned here, we saw our GPU usage reaching 100%, but the training of a single epoch lasted more time, starting with 1 hour and 10 minutes. Then, we tried installing Tensorflow from Anaconda, with the command:

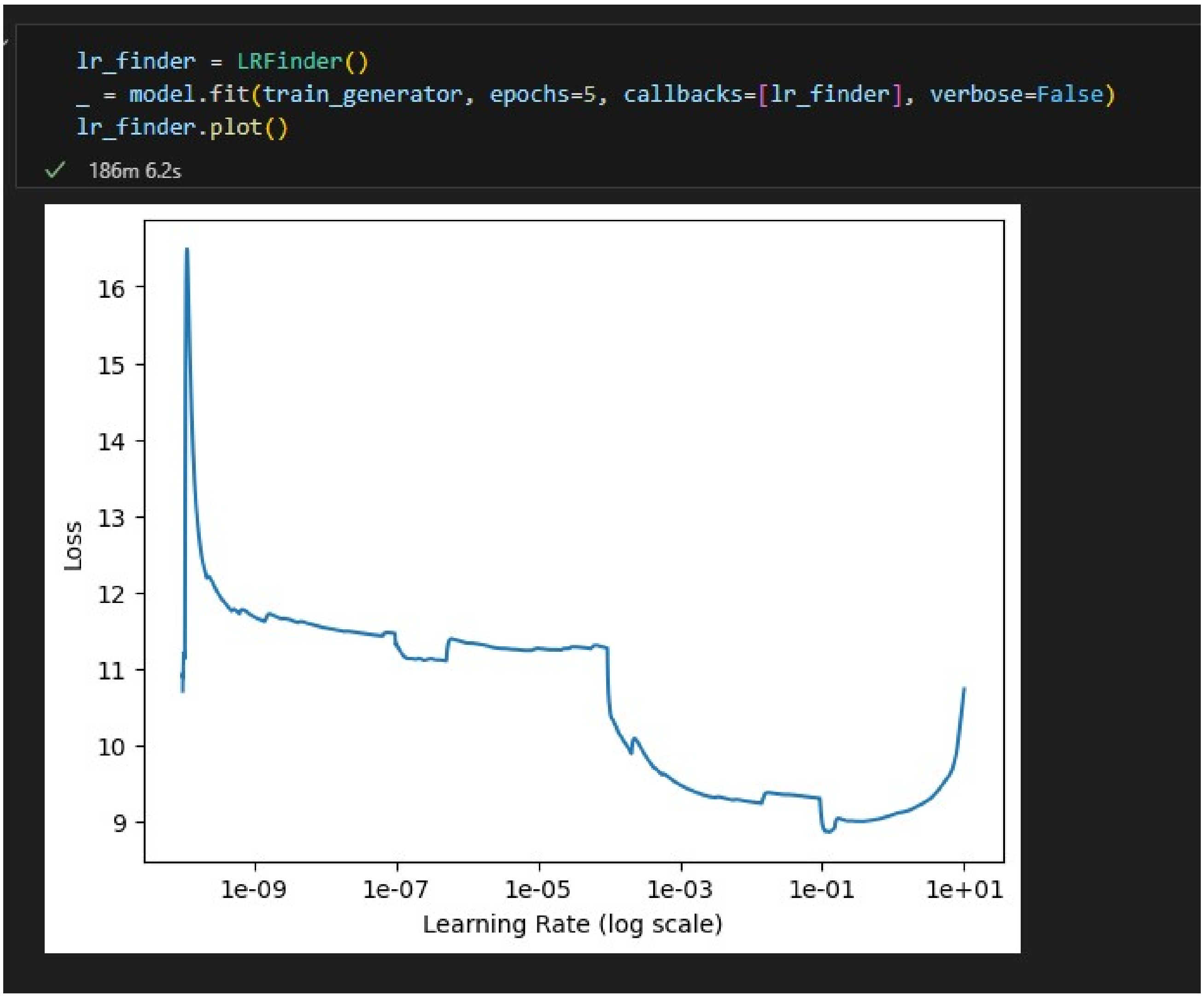
```
pip uninstall tensorflow
conda install tensorflow
```

We observed a slightly faster CPU performance, starting from 40 minutes per epoch instead of 50 minutes. It was indeed a performance, but not a 8.6x as mentioned here for ResNet50 model. Then, we tried to build Tensorflow from source on Windows, following the instructions mentioned here, but all attempts (and there were several of them) failed. Lastly, we ended up training our models in the cloud, with the use of Google Colaboratory (mostly) and Paperspace Gradient Console (less).

The CNN models were built using Tensorflow v2.13.0 API popular deep learning framework. For the five output layers of all models, we used a sigmoid activation function with a binary cross-entropy loss for the binary outcomes (gender, season, and subCategory) and a SoftMax activation function with sparse categorical cross-entropy loss (the encoded labels were integers and not one-hot-encoded labels) for the outcomes having more than two categories (usage and articleType). Also, for all models, we applied an Early Stopping Callback with a patience of 10 epochs and a Model Checkpoint Callback to store the weights of the best epoch with respect to the minimum validation loss. For the third and fourth models, we applied four variants for their architecture build:

1. The first variant included the Input Layer, the respective preprocessing input layer for each model (ResNet/EfficientNetV2), the respective frozen-layer pre-trained model, a flatten layer, a Rectified Linear Unit activation function dense layer, and the five output layers with a callback to reduce the Adam optimizer learning rate by 10, with a patience of 5 epochs of no decrease in validation loss. The range of best learning rates to start with and end to were found with a custom best-learning-rate-

finder (LRFinder) function to be in the range [0.0001, 0.1], as shown in Figure 11. The models were set to train for 100 epochs and stopped due to early stopping callback at the 61st and 77th epoch respectively for ResNet50 and EfficientNetV2S.



```
lr_finder = LRFinder()
_ = model.fit(train_generator, epochs=5, callbacks=[lr_finder], verbose=False)
lr_finder.plot()
```

✓ 186m 6.2s

Figure 11 - Best Learning Rates

2. The second variant was exactly the same with the first variant, with the only difference that the Adam optimizer learning rate remained stable at the default value of 0.001. The models trained for 60 and 80 epochs respectively, before stopping due to the Early Stopping Callback.
3. The third variant was exactly the same with the second, with the addition of 2 Lambda functions in the architecture, after the input layer and before the pre-trained model layer, that convert the images to grayscale and again to RGB (because the models accept only images with three channels and not one). This was included because we do not gain anything from the color for classifying an image if it is for Men/Women, it is Jeans or Trousers, Topwear/Bottomwear, Formal/Casual and if it is to be worn in the summer or in the winter. Even a person with color deficiency can recognize these features. The models trained for 72 and 77 epochs respectively, before stopping due to the Early Stopping Callback.
4. The fourth variant was exactly the same with the third, with two exceptions. Firstly, the unfreezing of the top n layers of the pre-trained model (actually achieving this way transfer learning) in order to fine-tune the pre-trained model on our dataset. For the ResNet50 pre-trained model we unfreeze the top 8 out of the 176 layers and for the EfficientNetV2S we unfreeze the top 64 out of the 514 layers.

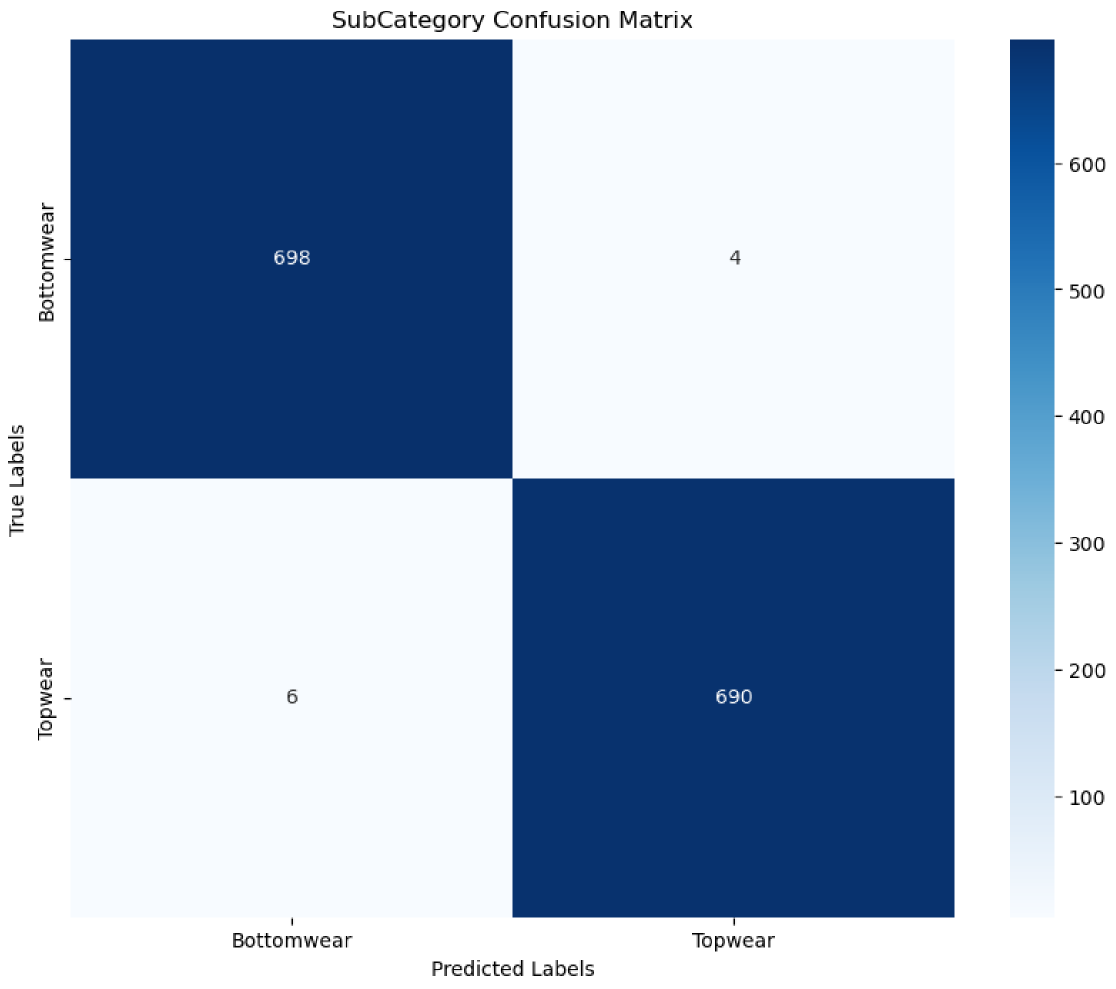
Secondly, we added a Dropout layer with a 0.2 rate before the ReLU activation function for regularization. The models trained for 25 and 35 epochs respectively, before stopping due to the Early Stopping Callback.

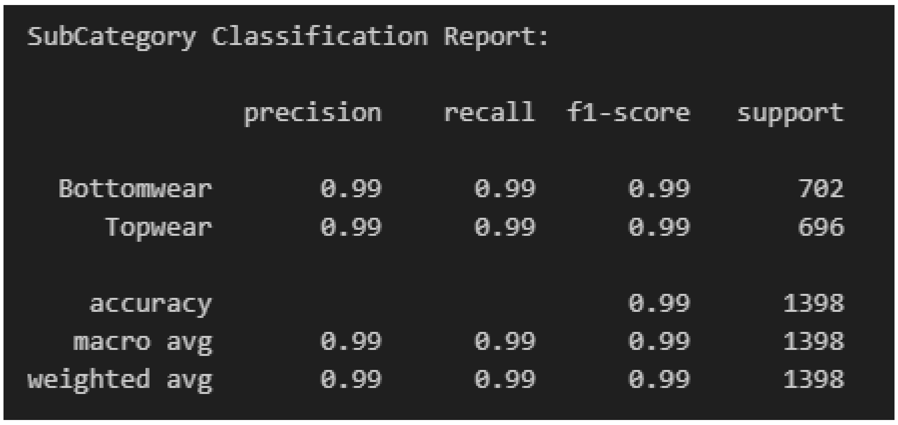
Results & Quantitative Analysis

Out of the forementioned 4 variants, the ResNet50 model achieved better results with grayscale images with all layers frozen i.e., the third variant and the EfficientNetV2S model achieved better results with grayscale images with top layers not frozen i.e., the fourth variant. The latter model was also the best out of all experiments achieving 96.4% accuracy in the training dataset, 90% in the validation and 90.7% in the test dataset. Because, the models are multi-output, the accuracy, loss, and all other metrics are presented per output. For report-space's sake, out of the 4 models and variants we will only present the best variant of the best model's metrics per output. For more details, readers are encouraged to view the 'b_Sklearn_Linear_Models' and 'c_Tensorflow_CNN_Models' Jupyter Notebooks.

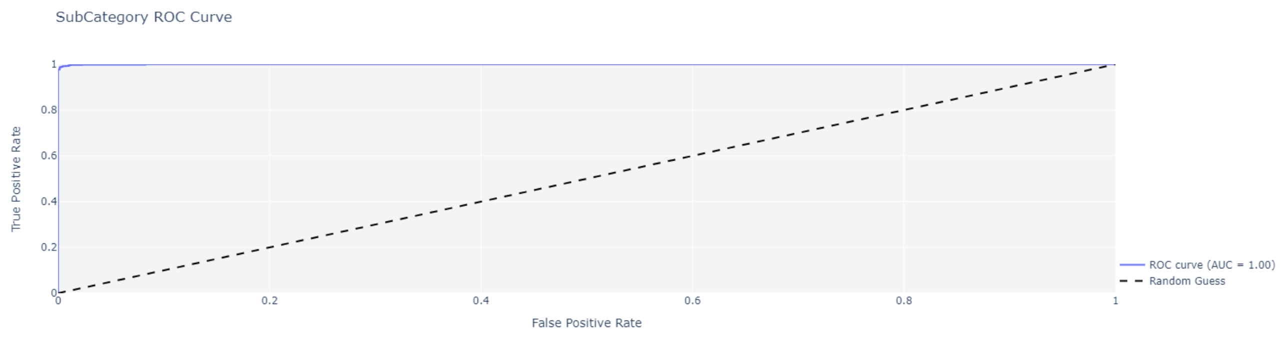
The quantitative results of the best model (EfficientNetv2-S Model Grayscale with Trainable Top Pre-Trained Layers):

<u>SubCategory output</u>:

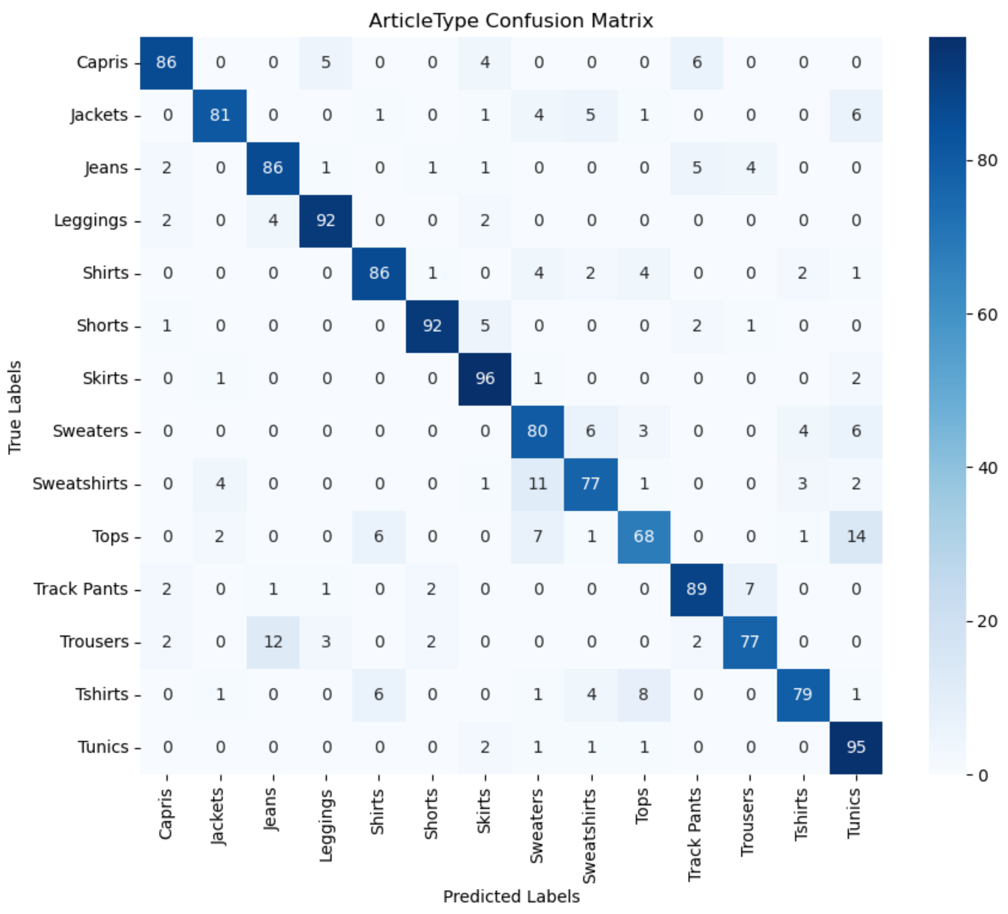




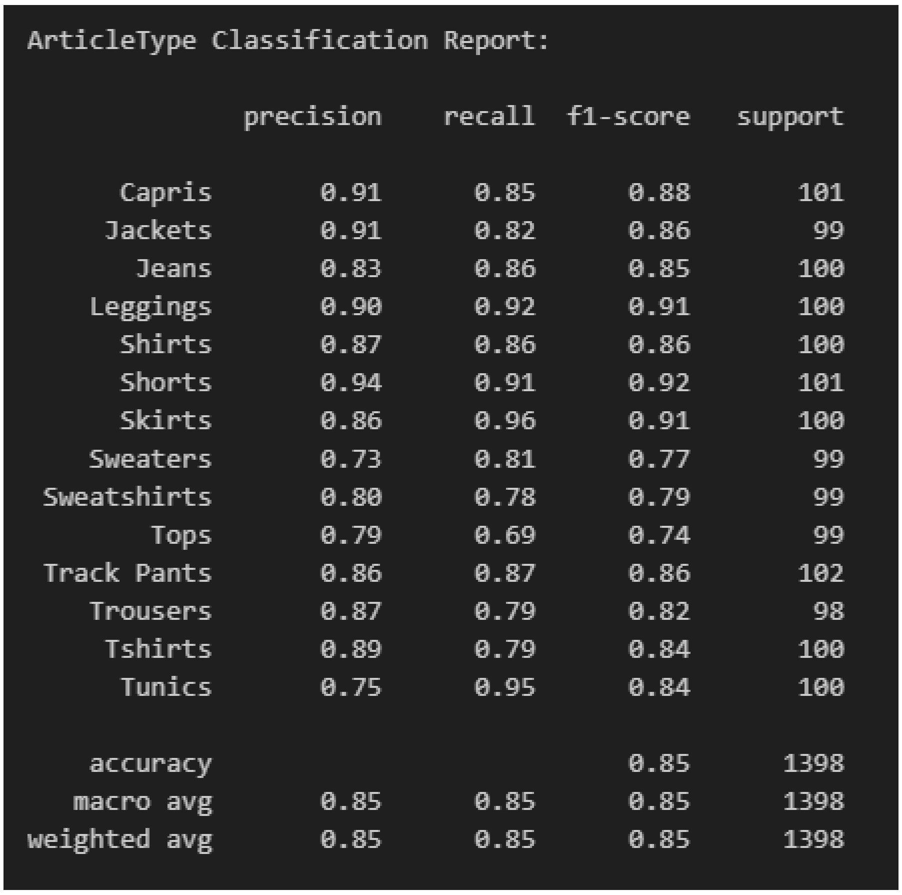
```
SubCategory Classification Report:

              precision    recall  f1-score   support

  Bottomwear       0.99      0.99      0.99       702
     Topwear       0.99      0.99      0.99       696

    accuracy                           0.99      1398
   macro avg       0.99      0.99      0.99      1398
weighted avg       0.99      0.99      0.99      1398
```



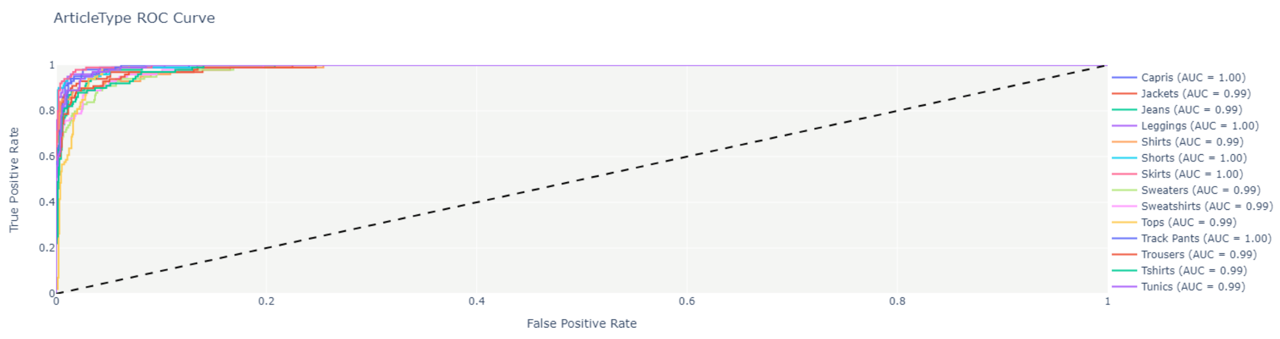
<u>ArticleType output</u>:



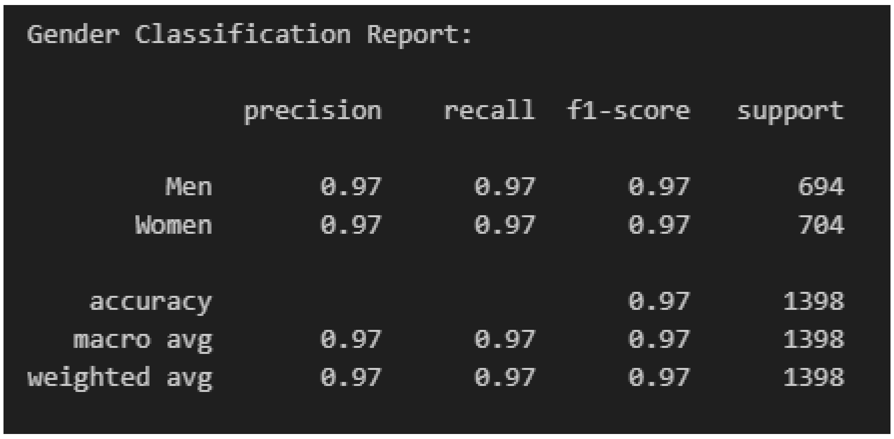
	Capris	Jackets	Jeans	Leggings	Shirts	Shorts	Skirts	Sweaters	Sweatshirts	Tops	Track Pants	Trousers	Tshirts	Tunics
Capris	86	0	0	5	0	0	4	0	0	0	6	0	0	0
Jackets	0	81	0	0	1	0	1	4	5	1	0	0	0	6
Jeans	2	0	86	1	0	1	1	0	0	0	5	4	0	0
Leggings	2	0	4	92	0	0	2	0	0	0	0	0	0	0
Shirts	0	0	0	0	86	1	0	4	2	4	0	0	2	1
Shorts	1	0	0	0	0	92	5	0	0	0	2	1	0	0
Skirts	0	1	0	0	0	0	96	1	0	0	0	0	0	2
Sweaters	0	0	0	0	0	0	0	80	6	3	0	0	4	6
Sweatshirts	0	4	0	0	0	0	1	11	77	1	0	0	3	2
Tops	0	2	0	0	6	0	0	7	1	68	0	0	1	14
Track Pants	2	0	1	1	0	2	0	0	0	0	89	7	0	0
Trousers	2	0	12	3	0	2	0	0	0	0	2	77	0	0
Tshirts	0	1	0	0	6	0	0	1	4	8	0	0	79	1
Tunics	0	0	0	0	0	0	2	1	1	1	0	0	0	95



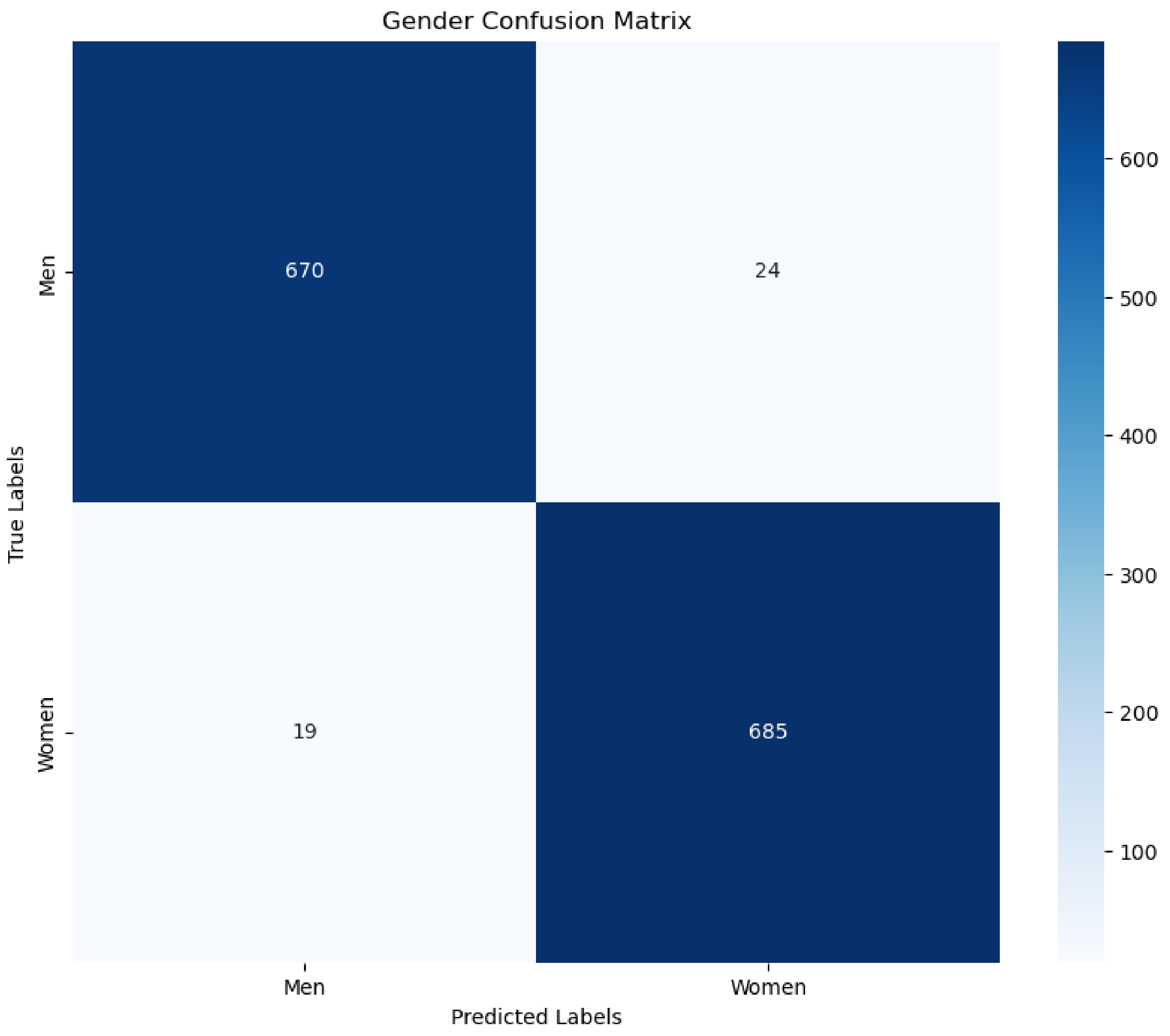
```
ArticleType Classification Report:

              precision    recall  f1-score   support

      Capris       0.91      0.85      0.88       101
     Jackets       0.91      0.82      0.86        99
       Jeans       0.83      0.86      0.85       100
    Leggings       0.90      0.92      0.91       100
      Shirts       0.87      0.86      0.86       100
      Shorts       0.94      0.91      0.92       101
      Skirts       0.86      0.96      0.91       100
    Sweaters       0.73      0.81      0.77        99
 Sweatshirts       0.80      0.78      0.79        99
        Tops       0.79      0.69      0.74        99
 Track Pants       0.86      0.87      0.86       102
    Trousers       0.87      0.79      0.82        98
     Tshirts       0.89      0.79      0.84       100
      Tunics       0.75      0.95      0.84       100

    accuracy                           0.85      1398
   macro avg       0.85      0.85      0.85      1398
weighted avg       0.85      0.85      0.85      1398
```

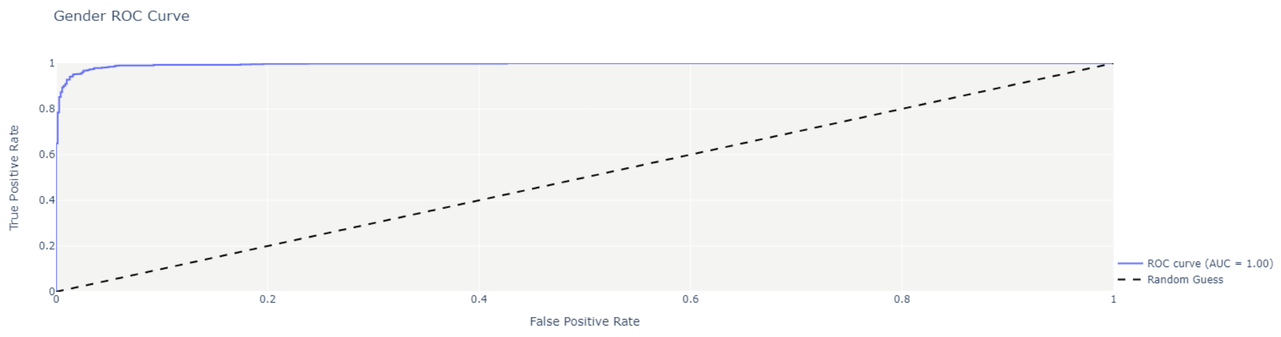


<u>Gender output</u>:

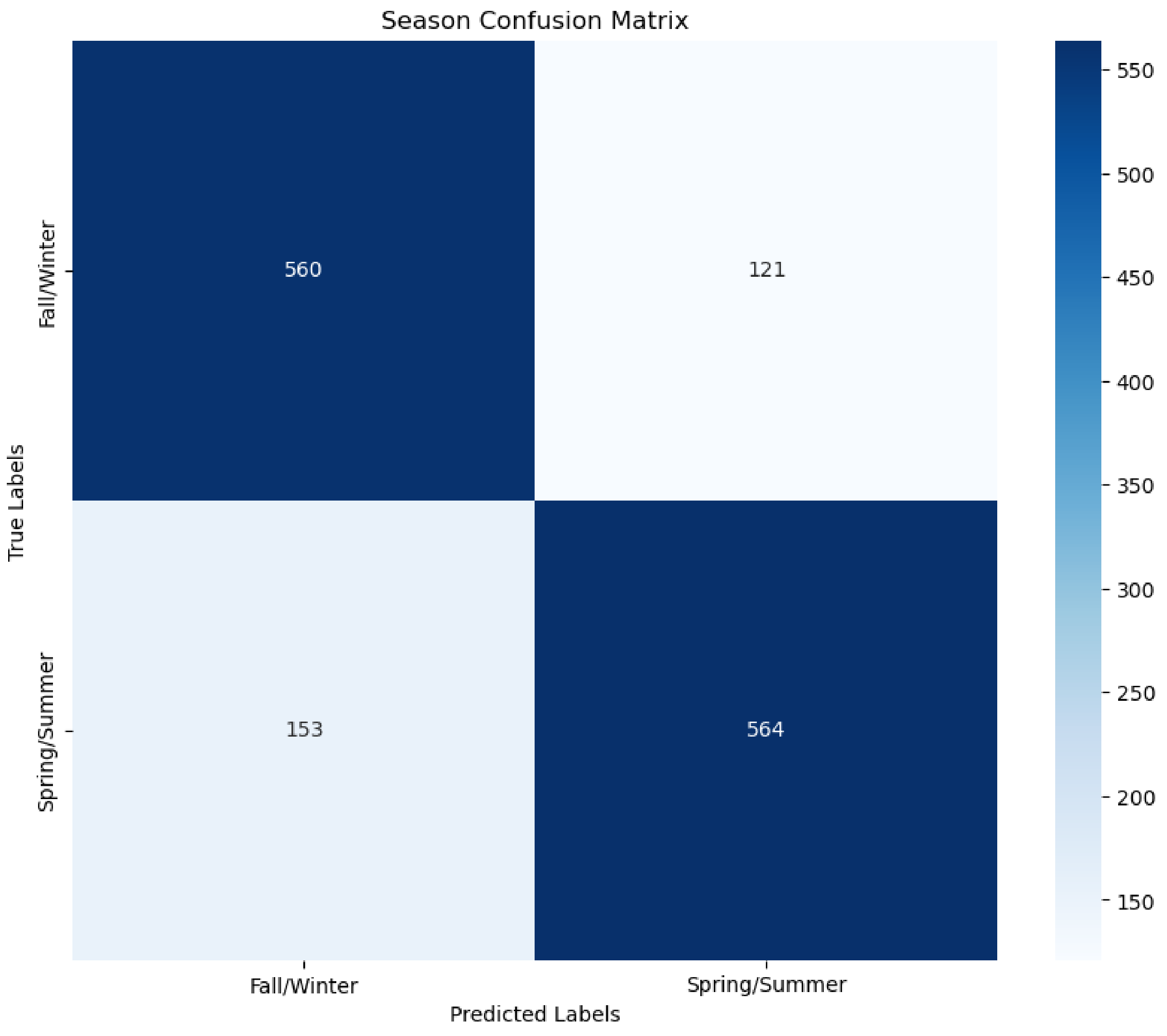


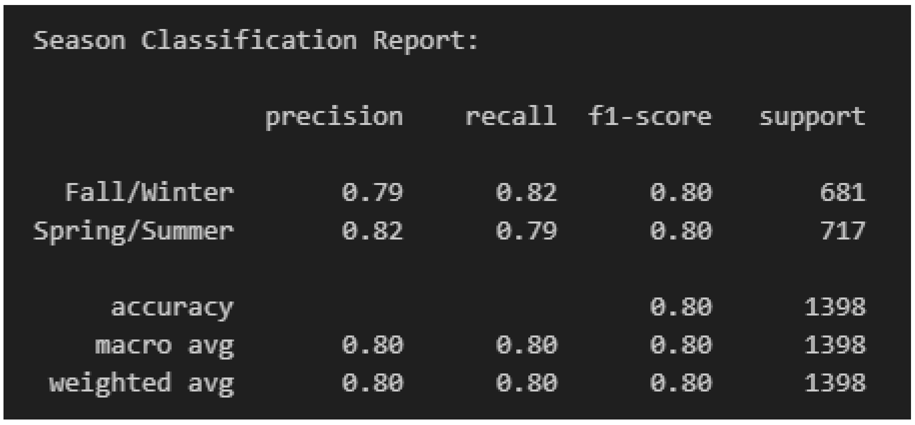
```
Gender Classification Report:

              precision    recall  f1-score   support

         Men       0.97      0.97      0.97       694
       Women       0.97      0.97      0.97       704

    accuracy                           0.97      1398
   macro avg       0.97      0.97      0.97      1398
weighted avg       0.97      0.97      0.97      1398
```


Gender Confusion Matrix
True Labels
Men
Women
670
24
19
685
Predicted Labels
Men
Women
600
500
400
300
200
100

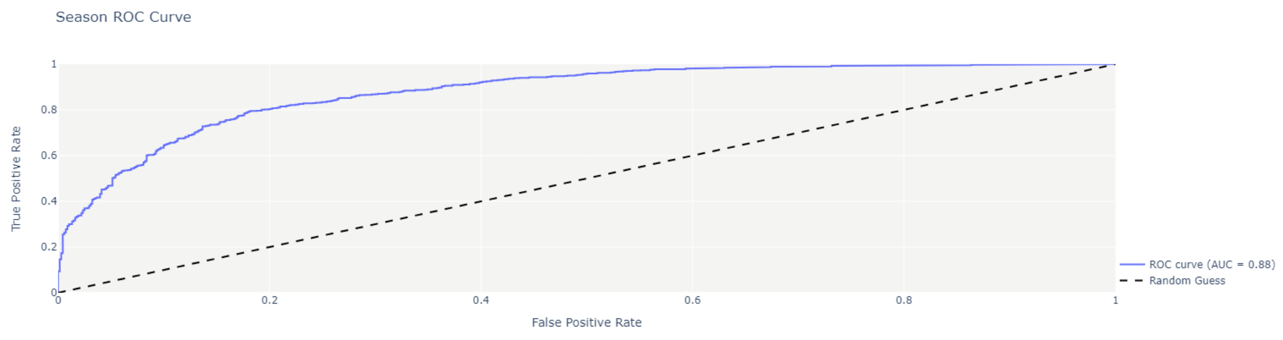

Gender ROC Curve
True Positive Rate
False Positive Rate
0
0.2
0.4
0.6
0.8
1
ROC curve (AUC = 1.00)
Random Guess

<u>Season output</u>:

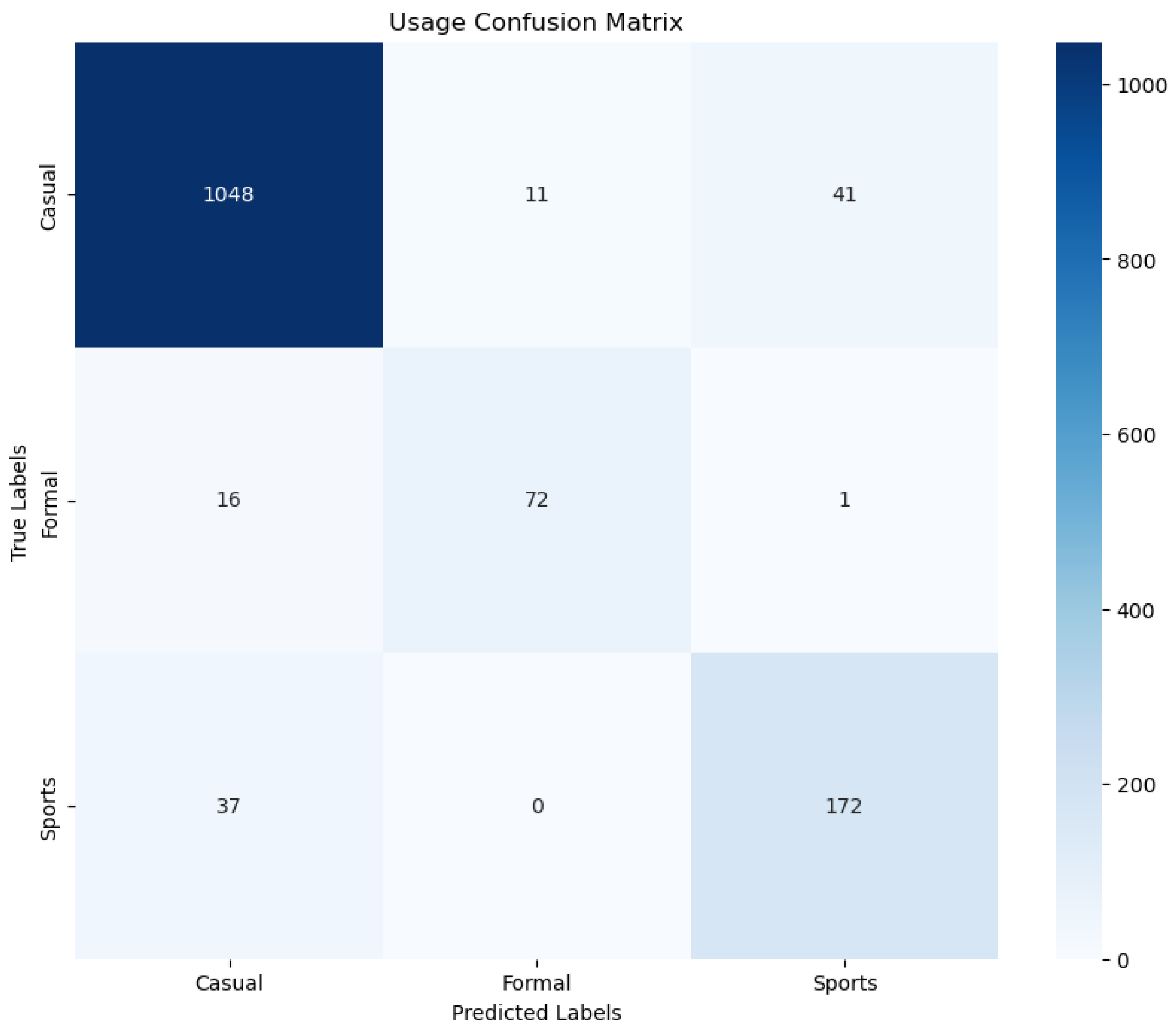


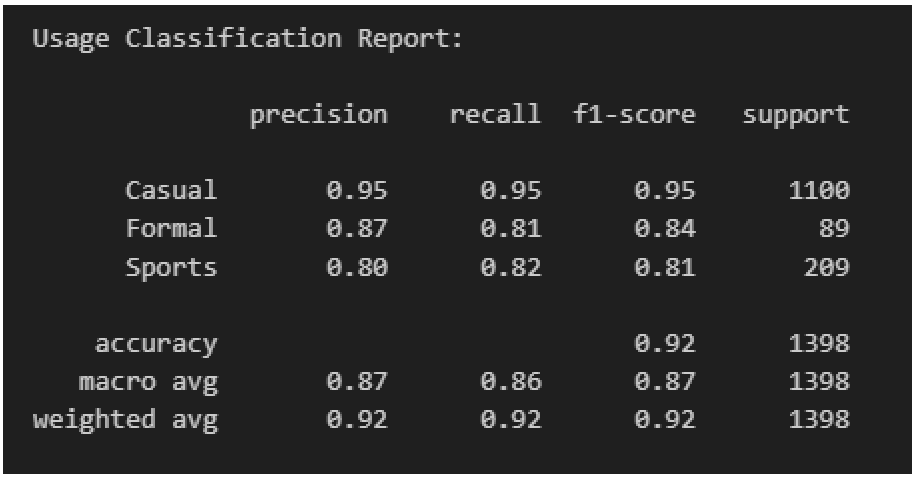


```
Season Classification Report:

               precision    recall  f1-score   support

  Fall/Winter       0.79      0.82      0.80       681
Spring/Summer       0.82      0.79      0.80       717

     accuracy                           0.80      1398
    macro avg       0.80      0.80      0.80      1398
 weighted avg       0.80      0.80      0.80      1398
```

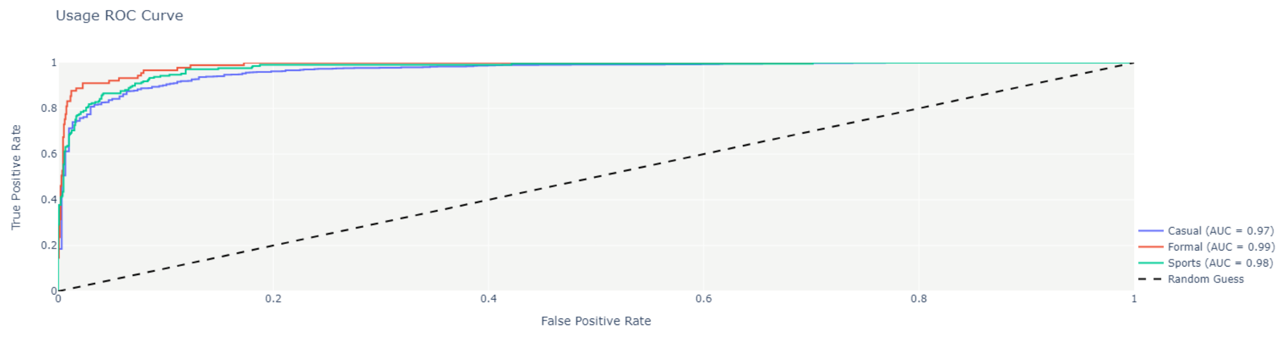


<u>Usage output</u>:



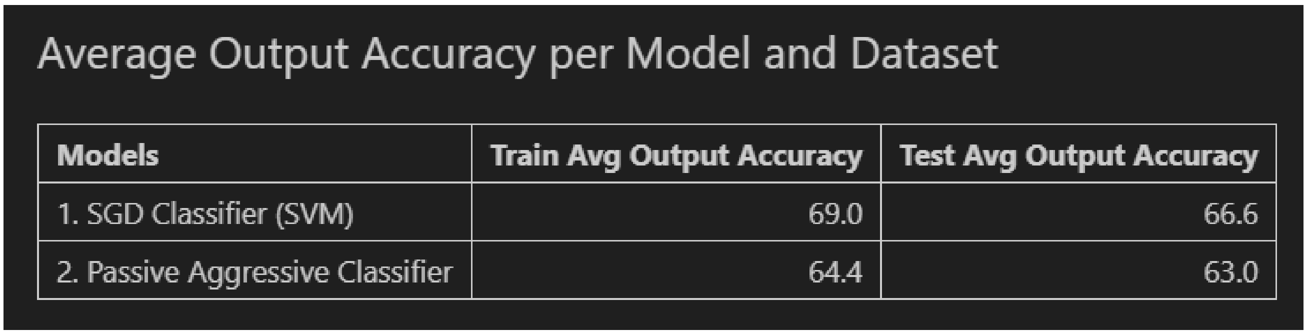


```
Usage Classification Report:

              precision    recall  f1-score   support

      Casual       0.95      0.95      0.95      1100
      Formal       0.87      0.81      0.84        89
      Sports       0.80      0.82      0.81       209

    accuracy                           0.92      1398
   macro avg       0.87      0.86      0.87      1398
weighted avg       0.92      0.92      0.92      1398
```



For a brief explanation of the confusion matrix, classification report, ROC curve and AUC (Area Under the Curve) metrics, readers are encouraged to read the respective section in the appendix.

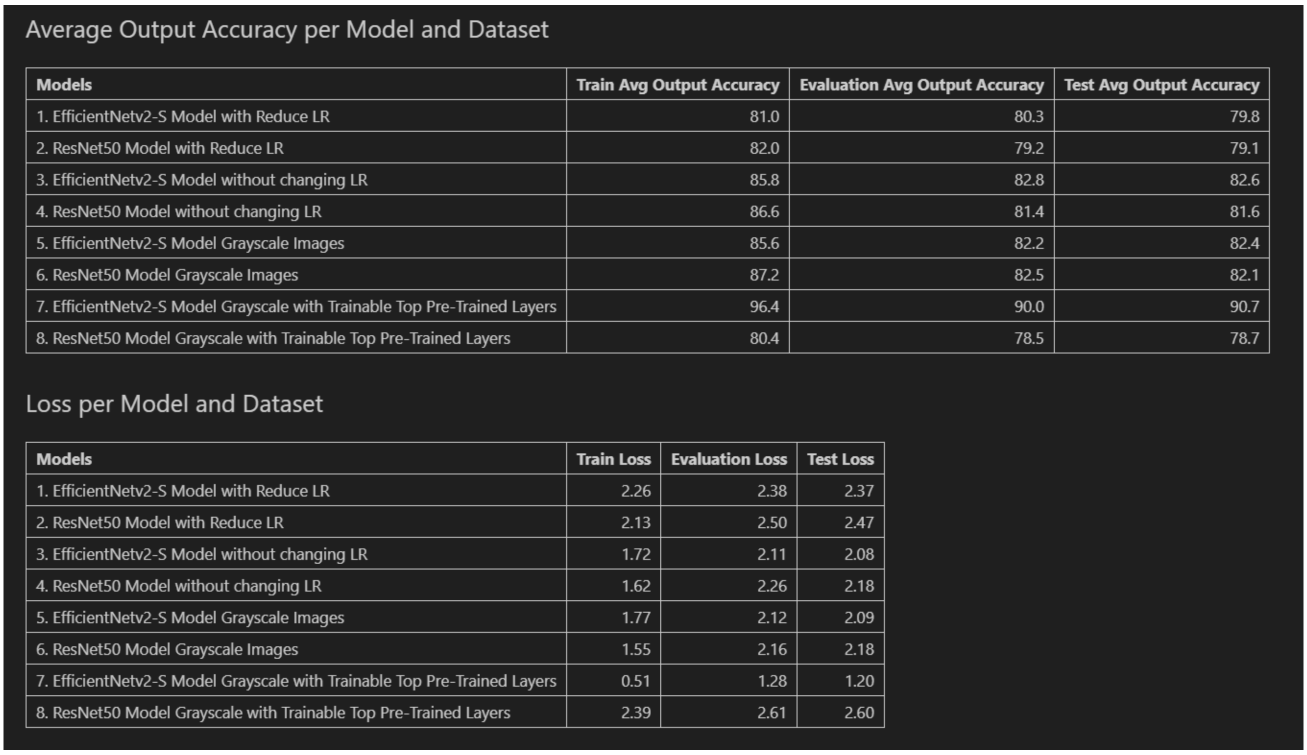
The summary of all models and variants accuracy and loss is presented below:



Average Output Accuracy per Model and Dataset

Models	Train Avg Output Accuracy	Test Avg Output Accuracy
1. SGD Classifier (SVM)	69.0	66.6
2. Passive Aggressive Classifier	64.4	63.0

Figure 12 - Average Output Accuracy per Model and Dataset for sklearn models



Average Output Accuracy per Model and Dataset

Models	Train Avg Output Accuracy	Evaluation Avg Output Accuracy	Test Avg Output Accuracy
1. EfficientNetv2-S Model with Reduce LR	81.0	80.3	79.8
2. ResNet50 Model with Reduce LR	82.0	79.2	79.1
3. EfficientNetv2-S Model without changing LR	85.8	82.8	82.6
4. ResNet50 Model without changing LR	86.6	81.4	81.6
5. EfficientNetv2-S Model Grayscale Images	85.6	82.2	82.4
6. ResNet50 Model Grayscale Images	87.2	82.5	82.1
7. EfficientNetv2-S Model Grayscale with Trainable Top Pre-Trained Layers	96.4	90.0	90.7
8. ResNet50 Model Grayscale with Trainable Top Pre-Trained Layers	80.4	78.5	78.7

Loss per Model and Dataset

Models	Train Loss	Evaluation Loss	Test Loss
1. EfficientNetv2-S Model with Reduce LR	2.26	2.38	2.37
2. ResNet50 Model with Reduce LR	2.13	2.50	2.47
3. EfficientNetv2-S Model without changing LR	1.72	2.11	2.08
4. ResNet50 Model without changing LR	1.62	2.26	2.18
5. EfficientNetv2-S Model Grayscale Images	1.77	2.12	2.09
6. ResNet50 Model Grayscale Images	1.55	2.16	2.18
7. EfficientNetv2-S Model Grayscale with Trainable Top Pre-Trained Layers	0.51	1.28	1.20
8. ResNet50 Model Grayscale with Trainable Top Pre-Trained Layers	2.39	2.61	2.60

Figure 13 - Loss and Average Output Accuracy per Model Variant and Dataset

Qualitative & Error Analysis

For the last variant of our models, we observed signs of overfitting. Because of these signs, we specified to the Model Checkpoint Callback to store each best epoch without overwriting the previous. In the figures below, it can be observed that as epochs pass, the model tends to overfit on the training data, thus not being able to generalize on new unseen data. Thus, for the ResNet50 model the 3rd epoch's weights were loaded as the best and for the EfficientNetV2S model the 19th epoch's weight were loaded as the best.

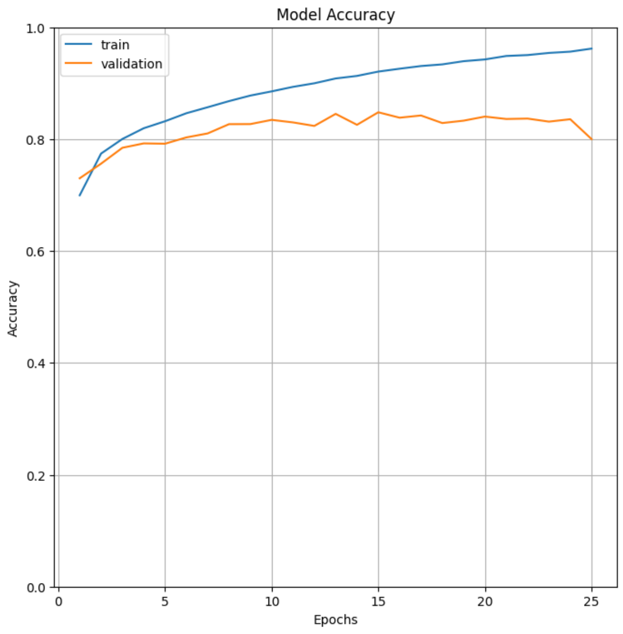


Figure 14 - ResNet50 Model Average Accuracy per epoch

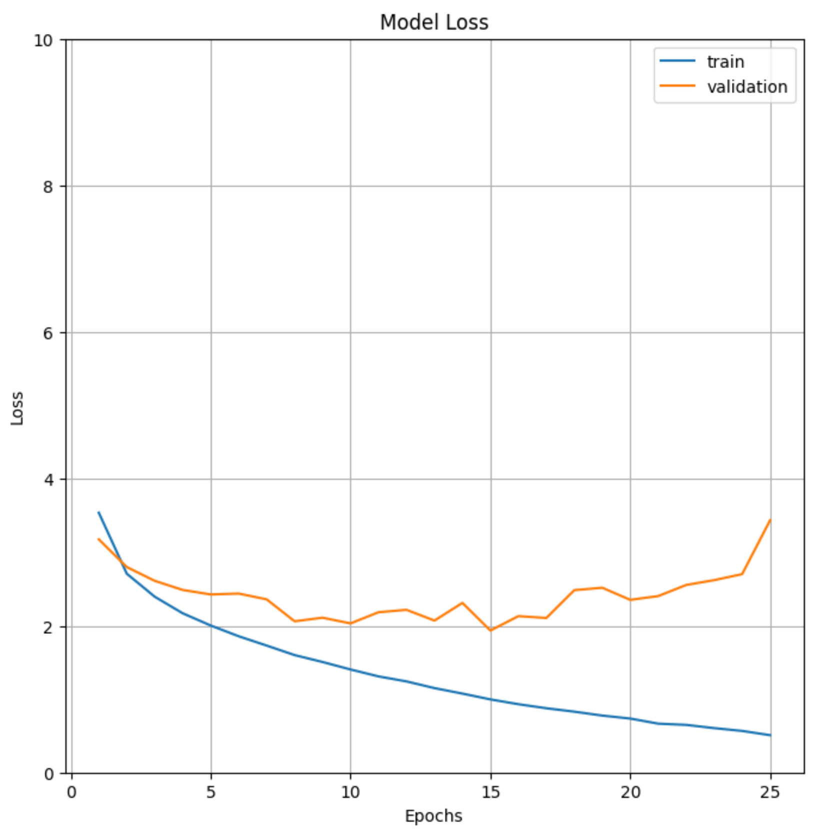


Figure 15 - ResNet50 Model Loss per epoch

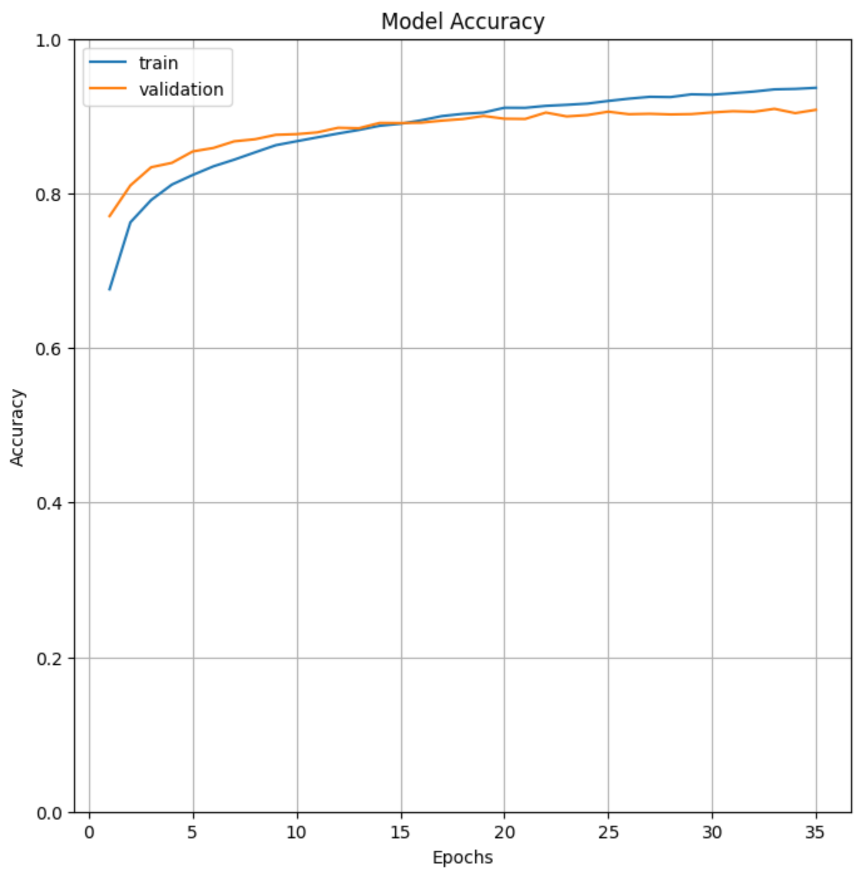


Figure 16 – EfficientNetV2S Model Average Accuracy per epoch

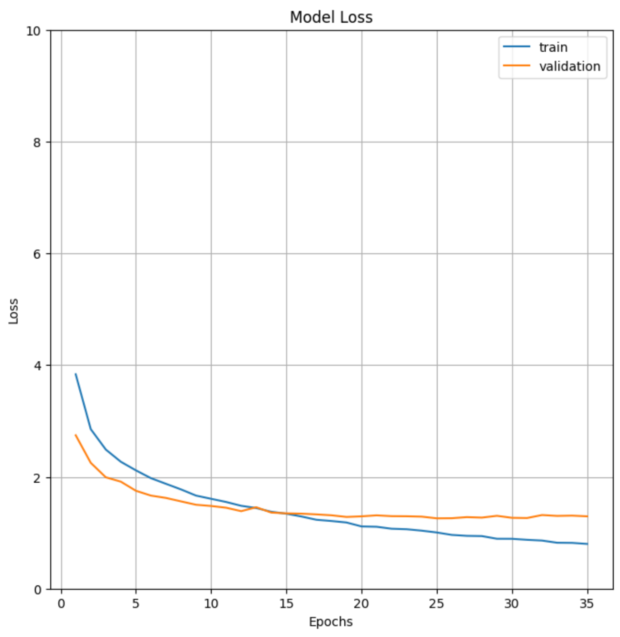


Figure 17 - EfficientNetV2S Model Loss per epoch

Generally, the best variant model has good performance on classifying clothes on the five forementioned output labels. We can observe a loss in its performance (80% on the test dataset) for the season output. That's maybe because of the intermediate seasons (Fall and Spring) that clothes have a thicker fabric but can be with short or no sleeves. Also, the lack of data for these seasons and the manual update of the season column from 4 labels (4 year's seasons) to 2 labels (2 fashion runway seasons) can be the explanation to this misclassification rate.

Discussion, Comments/Notes and Future Work

From the best model, we built an application utilizing Streamlit easiness of creating such. Readers of this report in order to use the application created just have to install Streamlit package and run the app via executing in terminal:

```
pip install streamlit
cd path/to/folder
streamlit run d_Streamlit_app.py
```

The app offers the following functions to its users:

1. View your DressMeUp wardrobe clothes (View My Wardrobe option)
2. Insert new articles to your wardrobe, either by URL path (single-insert) or by local files (batch-insert) (Insert New Clothes option).
3. Update article's features values or delete articles in your DressMeUp wardrobe (Update/Delete Clothes option).
4. Identify article's color by double-clicking on a point in the photo (Click & Update Color option).
5. Make combinations from the clothes in your DressMeUp wardrobe or from an online article and the clothes in your wardrobe (Combine Clothes option).
6. Plan your daily occasions (Daily Planner option).
7. Find the percent of clothes usage in your wardrobe's combinations (Clothes Usage Statistics option).
8. Delete the whole DressMeUp wardrobe (Delete My Wardrobe option).

Streamlit does not provide, so far, a way to construct an Android/Apple application. Because we understand that users prefer to take photos of their clothes and use a mobile app that is always with them, in the near future, we plan to create a mobile application. Until then, we have created a mockup screen using proto.io, as seen in the figure below.

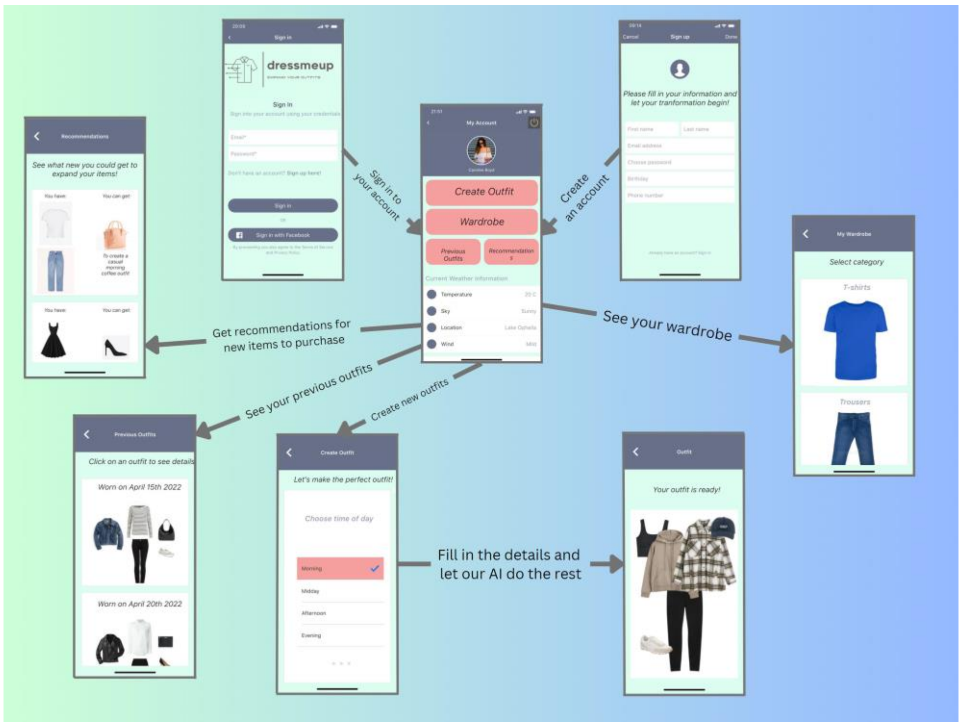
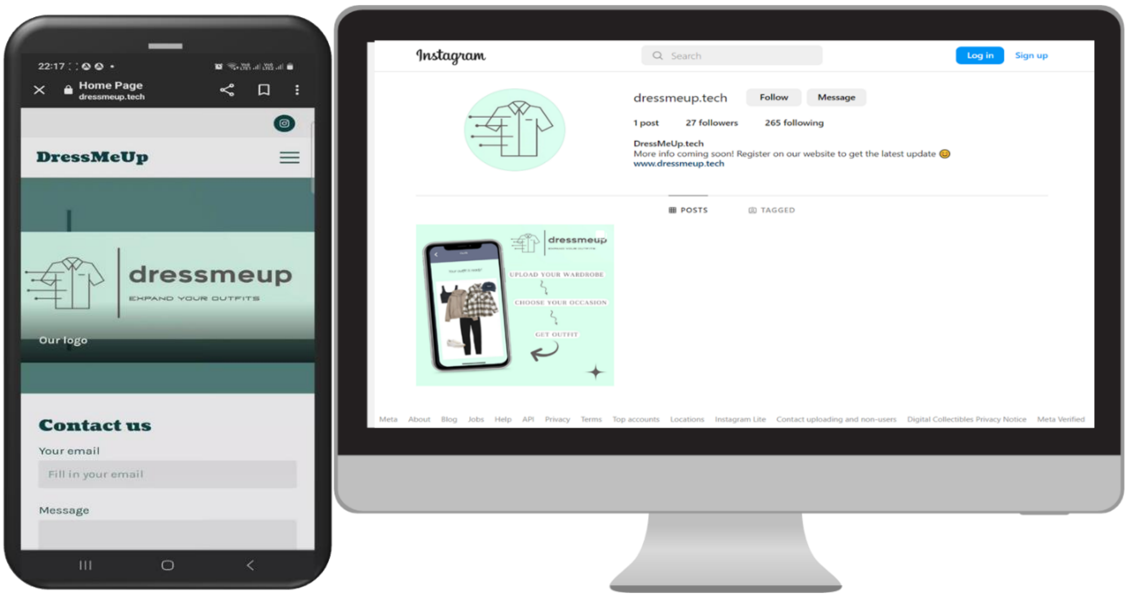


Figure 18 - Mobile Application Mock-up Screen

We also registered with the use of “Papaki” web hosting provider, the use for 1 year of the following domains: dressmeup.tech and dressmeup.site. Then, we issued a web hosting packet to accommodate the former domain. Then, with the use of “Papaki” for a month we built our web page and for a month we bought an SSL package in order to provide a secure socket layer to our webpage visitors to ensure them that they are safe when entering their contact information. Our webpage is https://dressmeup.tech, but was available for anyone to visit until 15/09/2023, when the Website Builder Packet expired. The site was built for educational purpose.

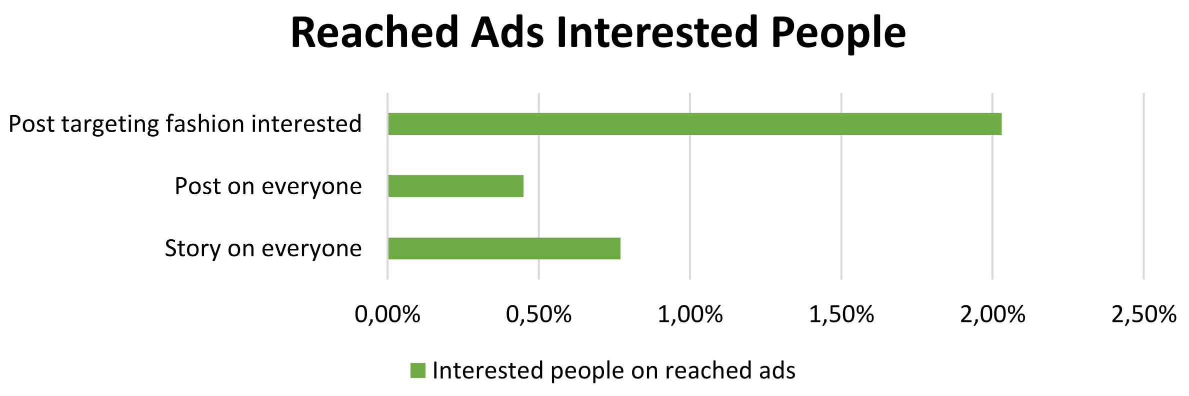


We also created an Instagram account (https://www.instagram.com/dressmeup.tech), in order to be able to advertise our company to future customers, link them to our web application and webpage, post news and features of the application and in general to familiarize the market with our business idea. The goal of the forementioned procedure was to track the market's interest in our business idea by advertising the features of our application (mock-up screens) and persuading future customers that are socially active in Instagram (the most popular social media for younger aged people) to land to our website landing page and express their interest for the application through an e-mail subscription (the e-mail with the message would be sent to one of our e-mails – tested that it was working).

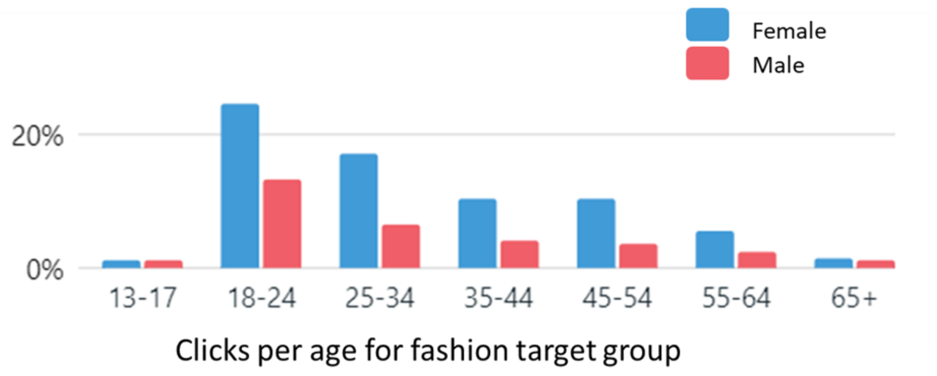
Customer Validation was done through Instagram advertisements. We created a post to evaluate the interest in our business idea. We did 3 types of advertisement campaign: an Instagram Story for no targeted audience, an Instagram Post for no targeted audience and an Instagram Post for targeted fashion-interested audience. The results are shown in the table below (Table 1).

Advertisement Type	Reached People	People clicked	Interest % of clicks per reach
Instagram Story to everyone	1237	9	0.73%
Instagram Post to everyone	1110	5	0.45%
Instagram Post to fashion-interested	837	17	2.03%

Table 1 - Results of Advertisement Campaign



The interest in our business idea increased for fashion-interested people. We can observe that, for the fashion-interested people that clicked our advertisement across all age groups, the female customer segment presented a larger attraction towards our business idea and gave back more clicks than the male customer segment. In particular, the age group of 18-24 years old introduced a larger interest in our application and broke the 20% barrier of clicking back the ad and landing to our Instagram account in order to get more information. However, from all people that clicked any of the forementioned advertisement campaigns created, none expressed their interest via filling their e-mail in our website's landing page.



In the appendix, it is included a Business Model Canvas. Readers are encouraged to view what's our thoughts about this business idea, if it is viable, the potential customer segments, etc.

Finally, in the near future, we also plan to train an Image Segmentation Model to segment products identified. This will also help the K-Means algorithm to be applied not to a fixed section of the images (that is applied so far), but to the whole product and extract the most dominant (base) color out of it.

Members/Roles

Our team is comprised of:

1. Konstantinos (Georgios Konstantinos) Vlassis, who will be our Chief Executive Officer. He holds a B.Sc. from the Management Science and Technology Department of Athens University of Economics and Business. Konstantinos has worked for 2 years as an IT Auditor for an auditing firm in Athens, Greece. His skills include good team coordination capabilities, planning, understanding of vertical systems integration and problem solving.
2. Lefteris (Eleftherios Efthymios) Souflas, who will be our Chief Technology Officer. He holds a B.Sc. from Military Science and Technology Department of Hellenic Army Academy and a Semester Military Education Certification from Hellenic Army Computer Programming School. Lefteris is currently working as a Database Administrator in the Hellenic Army Informatics Support Centre, since 2021. His skills include IT software development and maintenance, planning, quick decision-making and working under highly demanding environments.
3. Dimitris Tsapatsaris, who will be our Chief Sales Officer. He holds a B.Sc. in Mathematics and combines a mathematics background with previous sales experience. With a keen analytical mindset and a history of successful sales roles, Dimitris is poised to drive revenue and forge valuable industry relationships post-launch.
4. Manolis (Emmanouil) Sakellaris, who will be our Chief Financial Officer. He holds a B.Sc. in Economics from University of Piraeus. Manolis works as an Application Consultant in Diastasys, an Innovative Insurance Software Company, since 2022. His skills include requirement analysis, consulting, presentational efficiency and teamwork.

Bibliography

Datagen. (2023, August 30). *ResNet-50: The Basics and a Quick Tutorial.* Retrieved from Datagen: https://datagen.tech/guides/computer-vision/resnet-50/

Hasty. (2023, August 30). *Efficient Net.* Retrieved from Hasty: https://hasty.ai/docs/mp-wiki/model-architectures/efficient-net

He, K., Zhang, X., Ren, S., & Sun, J. (2023, August 30). *Deep Residual Learning for Image Recognition.* Retrieved from arXiv: https://arxiv.org/pdf/1512.03385.pdf

Mahmoud, R. (2023, August 25). *Fashion Seasons*. Retrieved from Retail Dogma: https://www.retaildogma.com/fashion-seasons/

Neo4j. (2023, June 12). *Neo4j Docs*. Retrieved from Neo4j: https://neo4j.com/docs/cypher-manual/current/execution-plans/shortestpath-planning/

Scikit-Learn. (2023, August 30). *1.1. Linear Models*. Retrieved from Scikit-Learn: https://scikit-learn.org/stable/modules/linear_model.html#passive-aggressive

Scikit-Learn. (2023, August 30). *sklearn.linear_model.SGDClassifier¶*. Retrieved from Scikit-Learn: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

Tan, M., & Le, Q. (2023, August 30). *EfficientNetV2:SmallerModelsandFasterTraining.* Retrieved from arXiv: https://arxiv.org/pdf/2104.00298v3.pdf

Tsang, S.-H. (2023, August 30). *Review — EfficientNetV2: Smaller Models and Faster Training*. Retrieved from Medium: https://medium.com/aiguys/review-efficientnetv2-smaller-models-and-faster-training-47d4215dcdfb

Appendix

1. Brief explanation of the confusion matrix, classification report, ROC curve, and AUC (Area Under the Curve):

<u>Confusion Matrix:</u>

A confusion matrix is a table used in classification tasks to evaluate the performance of a machine learning model. It displays a summary of the model's predictions and actual class labels for a given dataset. It is typically organized into four quadrants: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). These elements help assess how well the model is performing in terms of correctly classifying instances and identifying errors. For example, TP represents correctly predicted positive instances, while FN represents instances that were actually positive but predicted as negative.

<u>Classification Report:</u>

A classification report is a summary of various performance metrics for a classification model. It typically includes key metrics like precision, recall, F1-score, and support for each class or category in a multi-class classification problem. Precision measures the accuracy of positive predictions, while recall measures the ability of the model to correctly identify all positive instances. The F1-score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance. The classification report helps you understand the model's performance for each class, highlighting strengths and weaknesses.

<u>ROC Curve (Receiver Operating Characteristic Curve):</u>

The ROC curve is a graphical representation used to evaluate the performance of a binary classification model. It plots the true positive rate (sensitivity) against the false positive rate (1-specificity) for various thresholds used to make classification decisions. The ROC curve visually illustrates how well the model distinguishes between positive and negative classes. A perfect model has an ROC curve that passes through the upper-left corner, indicating high sensitivity and low false positive rate. The area under the ROC curve (AUC) quantifies the model's overall ability to discriminate between the classes. A higher AUC value (closer to 1) indicates better performance.

<u>AUC (Area Under the Curve):</u>

The AUC is a scalar value that quantifies the overall performance of a binary classification model based on its ROC curve. It represents the probability that the model will correctly classify a randomly chosen positive instance higher than a randomly chosen negative instance. An AUC value of 0.5 suggests that the model performs no better than random chance, while an AUC value of 1 indicates perfect discrimination between the classes. AUC is a useful metric for comparing different models or selecting the best threshold for making classification decisions. In summary, these evaluation metrics, including the confusion matrix, classification report, ROC curve, and AUC, provide valuable insights into the performance of machine learning classification models, helping you assess their accuracy, precision, recall, and ability to discriminate between classes.

2. Business Model Canvas

2.1. Customer Segments

The potential customer segments for our idea are:

B2B:

- Fashion and clothing companies anywhere in the world, that want to embed the API in their websites and provide more functionalities to their website, making it more attractive for customers.
- The same type of companies that want the AI algorithm to propose recommendations for their clothes for matching the users' wardrobe.

B2C:

- The mass market who wants to save time and simultaneously organize their outfit for all their weekly occasions by simply using the main services.
- Younger users who are interested on exploring all possible combinations of their wardrobe clothes, buy new clothes and explore their own personalized style.
- Individuals that would like to earn money by selling their unused clothes or keep track of their clothes' usage through a period of time, find their unmatched, poorly matched or unused clothes and make profit from them by selling to other users.
- Lower-income people that would like to build their wardrobe with less money than it would regularly need by buying from other users instead of clothing stores.
- Fashion professionals (e.g., designers, bloggers, influencers etc.) that want to create different outfits for their customers - followers.

2.2 Value Propositions

The services that our idea will provide create the following types of value propositions:

- Further easiness of people's lives.
- Wardrobe organization.
- Lower or no-cost creation of outfit abundance.
- Utilization & exploitation of unused clothes.
- Users' revenue creation.
- Increase of company's attractiveness towards their customers.
- Increase/retainment of company's market share and revenues.

2.3. Channels and Customer Relationships

Our company communicates with and reaches its customer segments to deliver the forementioned value propositions through mobile (Android & Apple) application, web application and API from any partnered e-shop. All these channels are integrated through a user sign-in verification. And it will be a sign-in through Google or Facebook account, that all people use in their daily routine. From the forementioned delivery of value propositions channels, the most valuable and effective will be the one via the mobile application, because people tend to use mobile phones everywhere. It will be the easiest way to upload their photos of the clothes they want to add to the app's digital wardrobe or to sell to other app users via the phone's embedded camera and also, the AI recommendation algorithm will take advantage of the embedded in the phone GPS tracker to identify the location and give recommendations based on local weather.

To retain our customers, notifications through the mobile application (Android or Apple) for new services, offers and features are going to be provided. Also, the creation and maintenance of social media pages will contribute towards that point. For our B2B customers, personal assistance through a helpdesk environment is going to be provided which is going to expand to all tiers of IT Support, which will include physical contact if needed.

2.4. Revenue Streams

Our company will generate revenues from the following:

- The 10% of the price of each item sold by the users.
- An annual fee from all premium users, who will not pay 10% per item sold.
- A direct API monetization fee for the API use in the e-shop websites.
- A small proportion to the price fee from fashion companies for promoting their clothes in the recommendation algorithm.

Our application will start being used for free with no ads in it by users who want to organize their wardrobe and dressing occasions. While our recommendation algorithm proposes the least used clothes to be sold to other users of the application, users will be persuaded to sell their clothes and thus earn money, leading to our company also earning money from a small percent of the price each cloth will be sold. As users will get used to this habit of selling their unused wardrobe, they will want to spend less as a fee for using our app, so they will achieve that from having a premium account with annual fee. The payment of the users will be executed via Credit Card, Debit Card, or PayPal.

Fashion e-shop websites, that wish to provide more functionalities to their website, making it more attractive for customers, will pay a direct API monetization fee for a thousand calls of our API from users. Fashion companies that want our AI algorithm to propose recommendations for their clothes for matching the app users' wardrobe will pay a small proportion to price fee.

However, for the latter type of revenue to come there must firstly come the former, but the revenues created from our B2B customers is estimated to contribute more to our overall revenues.

2.5. Key Activities and Partnerships

The first priority activity that provides value to our company overall is the development and maintenance of our AI algorithm that matches clothes depending on all forementioned conditions and the development and maintenance of the API that is going to be provided to the fashion e-shops. Another key activity will be the design and maintenance of the sell and purchase platform through our app as well as the design and maintenance of the UI/UX of the web and mobile apps. Once our company acquires the first B2B customer, the creation and maintenance of the helpdesk department of the company with personnel, hardware and software will be a key priority also. The rest of the personnel is planned to work from home, so there will not be much activity for the facilities and infrastructure support. In order to make our users' life easier we must first investigate what are their needs. This can be done by taking small surveys from their app usage experience or via the social media accounts. The price costing for the annual premium user fee and the monthly B2B fees for the API and the recommendations is another key activity that will balance between customer attraction/retainment and healthiness/survival of our company.

It is not scheduled in our plans to have any kind of partnership with other partners. Our key suppliers will be:

- domain registration and web hosting providers like "Papaki",
- advertising companies like social media advertising (e.g., Instagram) and
- helpdesk software support providers (when decided that it is needed).

2.6. Cost Structure

Our Business Model fall onto the category of cost-driven Cost Structures, focusing on minimizing costs wherever possible and aiming at creating and maintaining the leanest possible Cost Structure, using low price Value Propositions and maximum automation.

As our Cost Structure do not include costs that vary proportionally with the volume of services produced (variable costs), the most important fixed costs inherent in our business model will be:

- Personnel salaries
- Application registration
- Domain Registration and Web Hosting
- Server acquisition
- Advertising
- Helpdesk software.

Given that the application will be developed and maintained with open-source software and that it will not need the creation of a large database, as the clothes of each user will be stored with the exploitation of client-side databases, our company will not spend any money on database servers. The most expensive key resources will be the servers used, the helpdesk software and the personnel salaries, whereas the most expensive key activity will be the advertising.