Taste Profile Generation in Vogue using Deep Learning

DSE5499 – PROJECT WORK

Submitted in partial fulfilment for the award of the degree

of

Master of Technology

in

Data Science and Engineering

by

KARTHEEK PALEPU (15MDS0031)

Guided

by

PROF. KUMARAN. U

School of Information Technology and Engineering

February, 2017

Abstract

Our project aims to build a Convolution neural network (CNN) to generate a taste profile for a user based on his likes and dislikes in the field of fashion (Vogue). We use this taste profile to do a recommendation of new arrivals in fashion. The main idea is to take an initial survey for the user on some group of fashion items to form a taste profile for the user. We then train a CNN on his profile that helps us in doing a supervised learning on various fashion items with a flag called like/dislike. We use only the images to train a network and then use this network to do further recommendation. The main reason for the use of CNNs are, they automatically construct abstract features like edges, faces ...etc. that we must hand craft and feed into the model. These abstract features are so generalized that they account for variance. As CNNs are computationally intensive we use a pre-trained CNN model called Inception by using Google Tensor Flow which was trained on over 100K images with over 1000 categories. We use this inception model to transfer the learning done on a previous session to a new training session. Once the learning process is completed we apply this learning on the set of new arrivals to classify and recommend them to that user.

TABLE OF CONTENTS

1. Introduction

The Taste Profile Generation engine is a machine learning platform that runs on top of a deep learning algorithm Convolution neural network. This platform helps in processing and performing all the tasks of classification, clustering and finally recommendation based on the classification and clustering data. Usually "Machine Learning" is a bag full of algorithms, where a specific algorithm gets trained on a data (in this case the data is in the form of images) so that it can figure out the patterns in the data and use those discovered patterns to come up with new results for the new data. Our platform uses the image only technique where just the image is used to generate taste profile for a user instead of entire meta data about the vogue item. The following concepts in machine learning are used for this platform

a) Classification

It is an approach of information filtering that uses the image pixel data (in our case data for each of 100x100 pixels) and use different classification algorithms that uses this labelled pixel data (labelling is done by the user either in survey or his previous purchase history) to do classification whether the user likes the item or not. All the classification algorithms use the raw pixel details to do the classification which is not the perfect way. Also, the loss in using such classification algorithms are we need to do a lot of pre-processing of images by resizing all the images to an equal dimension to help the functioning of algorithms. Thus, to overcome these pre-processing steps we end up in building a deep learning model called Convolution neural networks where these abstract features are built in default and no pre-processing is required. Once the model is built we use this model to classify the new arrivals.

b) Clustering

It is an approach of grouping the customers based on their taste profile. This step is done after the model is built. The grouping of customers is also called customer segmentation and is often done to achieve best results in a simple way. All the customers in a single group are observed to have similar behaviour thus helping us to make recommendations to the entire group. The recommendations done to the entire group are assumed to be standard for all the customers in the group.

c) Recommendation

This is the final and the most important task done in this platform. This either uses the individual taste profiles generated from the raw classification engine or uses the segmented cluster taste profiles generated from the clustering engine. If individual taste profiles are used we can classify on the new arrivals for each user and suggest the top-10 items to that user thus making this process slightly more computationally intense but, it ends up in achieving more crisp recommendations compared to the clustered recommendations. If clustered taste profiles are used, we can classify new arrivals just for the cluster thus reducing the computational work and ending up in giving decent recommendations.

1.1. Scope

The scope of this platform is to automate the world of Vogue by using the concepts of computer vision and deep learning algorithms. In a way making the imagination of "walking into an E-Commerce website and purchase a fashion item just by looking at the image and price and getting recommendations based on the visual impact of your previous purchase" true. This platform can be extended into any item in the e-commerce platforms and end up in achieving great results with less human effort.

1.2. Objective

The main objective of this platform is to run a recommender engine on top of a classification engine and clustering engine by using different sets of the machine learning algorithms mainly using the core concepts of computer vision and deep learning. The complete objective of this platform can be broken down into 3 different objectives which are interlinked and together achieve a platform supporting decision making in Vogue. The 3 sub objectives are for the different engines running in the platform to come up with a final recommendation. They are classification engine, clustering engine and recommender engine. Each has its own objectives that are contrast with each other but merges together ending up in achieving the final platform objective.

To describe the main functional objective, we will cover the individual objectives of each engine.

- The objective of the classification engine is to use the most advanced machine learning algorithms, computer vision and deep learning algorithms into the very fabric of an item as you choose. This aims to build a nearly human accurate AI that understands the pattern of your purchase of fashion items just by looking at the image.
- The objective of clustering engine is to build a set of groups based on their similarity in behaviour of purchases thus reducing the effort of maintaining individual profiles
- The objective of the recommender engine is to generate recommendations to the users based on their groups (if using the clustering engine) or based on their individual behaviours (if using just the classification engine).

1.3. Motivation

The idea of the project is originated from the following references - Handwritten digit recognizer [R3] - where any supervised algorithm is used to classify hand written digits and CNN outperformed all of them, ImageNet challenge [L4] - where images of size nearly 2TB with multiple categories are used for training and the model is used to predict on the unseen dataset to measure the accuracy - here CNN is able to achieve more than human level accuracy percentage of over 94%, Generic object recognition [R4] - where multiple objects are recognized inside an image and a caption is allotted to that image based on the detected objects. We collaborate all these ideas to implement an inception network (which was already trained on millions of images with over 1000 categories) from Google's tensor flow module where we can use this inception to retrain on the domain of fashion.

1.4. Problem Statement

Using this platform, we made the maintenance of online fashion shopping easy for all the e-commerce websites like Flipkart, Amazon, Myntra ... etc. To define the problem statement, we use the following notions:

- $n = number\ of\ fashion\ items$
- $u_i = user\ with\ id = "i"$
- $p_i = taste\ profile\ of\ user\ with\ id = "i"$
- $i_x = fashion\ item\ with\ id =$ "x"
- $r_{xi} = rating\ for\ the\ item$ "x" $given\ by\ user$ "i"
- $pr_{xi} = predicted\ rating\ for\ the\ item$ "x" given by user "i"
- $Root\ mean\ squared\ Error\ (RMSE) = \sqrt{\frac{(r_{xi} - pr_{xi})^2}{n}}$

Our problem statement aims to reduce the $RMSE$ by building a model that can give us a clear separation of the above mentioned two classes of images. In depth, an initial survey is conducted for a new user where a set of n fashion items are rated by each of the i users in a binary format saying whether they like it or not ($1\ or\ 0$). This survey data is used as labelled target with image as features for the classification engine. Then, a classification engine runs on the data generating the profile p_i for user i. The profile is tested on an unseen image set with labels to predict the value pr_{xi} and use original r_{xi} and pr_{xi} to measure the performance - in our case it is $RMSE$. The models after refined tuning resulting in less $RMSE$ is the model that is chosen as the final model that can be used for further recommendations.

2. Minimum System Requirements

2.1. Hardware Requirements

Operating System	Linux (Preferable)
Primary Memory	4 GB (min.)
Secondary Memory	100 MB (min.)

2.2. Software Requirements

Name	Version
Python	2.6.6
PHP	7.0.13
My SQL	5.0.12
D3 JS	4.2.2
GIT	2.9.0
Docker	1.12.2

2.3. Package Requirements

Software	Packages
Python	Numpy
Python	Pandas
Python	Scikit-learn
Python	Scipy
Python	Tensor flow

3. Existing System

The existing system for our current implementation is nothing but all the fashion e-commerce websites like Flipkart, Amazon, Myntra... etc. All the mentioned organizations when receive the stock of fashion asks its employees to manually enter the details of each item in the stock in database into their respective databases. If it is a T-shirt the details include colour, fit, pattern ... etc. If it is a trouser the details include colour, fit, half or 3/4 or full, material ... etc. Thus, the systems have complete product information in the form text inside tables in database. Thus, when the user enters their website they can provide filters on colours, fit, sizes to help him with selection of his interests. Also, they use the purchase history of the users and map them back to the data residing inside the tables to do recommendations. The recommendation process is done by using content based or collaborative filtering systems where the user to user similarity is measured and products are suggested or item to item similarity is measured and products are suggested. All this needs the data to be entered inside which is hectic task for any employee to do. Even the data is entered the tables by a specific data entry team, these recommendations obtained from the purchase history is not enough and still show uninterested products for the user because the recommendations does not get visually impacted by the user, they are completely based on the data entry and does not take the visual look of the item into consideration. Per a survey conducted by the e-commerce organizations, 80% of the users are assumed to purchase the product by looking at the product and the other 20% of the users go through the product details and then end up in buying the product (especially in the case of Vogue). This may result in a user going through entire details of the product, select the filters to search and still will be ended up in not buying anything. Thus, this procedure will increase the disappointment for the user resulting in loss of customer satisfaction.

4. Proposed System

Taking the drawbacks of existing system, we have tried achieving to build a system that automates the data entry part and ending up in giving a better and more powerful recommendations for the user. This proposal differentiates with the existing implementation in lot of aspects. In the existing implementation, a manual entry of aesthetics like colour, size for the vogue item is required. But, in this proposal we can directly skip the details entry and concentrate on the visual inspection which gives more insights. i.e. Using the concepts of computer vision, machine learning with deep learning we can study the image of the vogue item and store the abstract details of the item like colour, size, fit ... etc. All this is automated just by building machine learning models like SVM, Random Forest, Artificial Neural networks (ANN) and Convolution Neural networks (CNN). These models like SVM, Random Forest, ANN use the pixel details to understand the features but does not build any abstract features by themselves. Thus, we finally end up building a Convolution Neural network (CNN) on these images. They build abstract features for each image and understand the image close to human level accuracy. In our system, we can either ask the user (if the user is a new user) to take a survey for a set of images and classify them as like/dislike for building his taste profile or we can also use the previous purchase history (if the user is an existing user) to re-tune the CNN model and update his taste profile. We can then use this taste profile to do either customer segmentation or individual customer profile analysis. By doing the customer segmentation we can group the customers with similar taste and provide recommendations to individual groups rather than an individual. By doing the individual customer profile analysis we can analyse his taste profile and can come up with more crisp recommendations that suits for a customer in a better way. The latter technique needs high computational power to process individual taste profiles and high storage capacity to store the taste profiles. All these issues can be sorted by using a cluster of machines running on a cloud.

5. Literature Survey

Taste profile generation using deep learning is a completely new concept evolved to overcome the difficulties of human data entry. In this process, there is no manual human interference. In the industry, there are various critical evaluations done on the topic of taste profile generations and deep learning but not together. Some of them are:

a) **Amazon Recommendations [Item-to-Item Collaborative filtering] by Greg Linden, Brent Smith and Jeremy York from Amazon:** This paper is the base paper for the current existing systems. The main gap in this paper that is filled by our platform is that there is no requirement of human data entry about the specifications of each item. (although price and brand is mandatory.)

b) **Google News Personalization: Scalable Online Collaborative Filtering:** This paper uses the concept of sequence to sequence models to generate taste profiles for each user. They are also called deep learning models (Recurrent Neural networks) but they are done on sequential data like text. Our platform aims to apply the concept of deep learning in the field of images.

c) **The Netflix Recommender System: Algorithms, Business Value, and Innovation:** This paper aims to do recommendation of Netflix shows to different users. This system also achieves recommendations on a structural data where our platform achieves using unstructured data like images.

d) **ImageNet large scale visual recognition challenge, O. Russakovsky et al:** This papers use the concept of Convolution neural networks on different sets of images to do the process of classification. But, our project aims to use this classification to do a further recommendation.

e) **Learning hierarchical features for scene labelling (2013), C. Farabet et al. (LeCun):** This paper aims to generate a caption for each image by studying all the subjects in the image. In contrast, we study the subjects to generate taste profile of the user.

In all the above literatures, researchers define and explain the problem of recommendations using data in a structured format. They also use the concept of deep learning to enhance the process of image classification. To summarize the previous investigations to inform the readers, the current problem is integration of the two concepts. i.e. taste profile generation on images.

6. Core Concepts

6.1. Machine learning

Machine learning is the term used where computers can solve multiple problems by using the same algorithm with the ability to learn without being explicitly programmed. i.e. The machine learning term is used when an algorithm generalizes for a data without writing a specific code to the problem. i.e. Instead of writing the code, we feed the data to the generic algorithms and allow the algorithm to learn the patterns hidden inside the data. For example, the same algorithm can be used for fraud detection in insurance domain as well as the classification of handwritten digits. There are two kinds of Machine Learning Algorithms that are mostly used.

1) Supervised learning Algorithms

Supervised learning is a task of pattern extraction from a labelled training data. i.e. In training data, each point consists of set of attributes and its corresponding target label. The algorithm analyses and understands the underlying patterns in this training data which can be further used to classify the unseen data. These algorithms are mostly used for classification and regression problems. Some examples for supervised learning algorithms are SVM, Linear regression, K-Nearest Neighbours … etc. In the case of our platform we use Supervised learning algorithms to train on labelled image dataset to come up with taste profile for every user.

2) Unsupervised learning algorithms

In the field of data science or data mining, an Unsupervised learning algorithms are used to find the hidden structure on the data where there are no labels. These algorithms are mostly used for customer segmentation, clustering and so on... Since the samples given to the algorithm are not labelled, there is no error measure to evaluate a solution. Some examples for unsupervised learning algorithms are K-Means clustering, Hierarchal Clustering, DBSCAN … etc.

6.2. Classification

Classification is one of the top problems among Supervised learning algorithms. The main aim of classification is to predict the target class (can be binary or multi-class problems). In this process, a Supervised learning model is trained on a training labelled data and the model is used to predict the class on an unseen data. The model understands the underlying patterns that will give clear separation among the classes in the training data. It understands and memorizes the patterns observed for each class. There are so many classification algorithms that are used in the real world - Some of them are logistic regression, Support vector machines (SVM), decision trees, random forest, artificial neural networks, deep learning algorithms... etc. In the case of our platform we use SVM, Random forest, Artificial Neural Networks and Deep learning to train the labelled image data set and generate taste profiles for every user and come up with a decent recommender systems.

6.3. Recommender systems

A recommendation engine is a technique that is used to filter the items by predicting the ratings of a user based on his similarity with other users or other items he has rated. It is a global solution for the problem of marketing the new items to the existing users to improve the sales. It is currently the trending technique used in all online shopping websites. It empowers the users by giving right to them for choosing a product that they may like in near future. The process of recommendation happens in usually two ways:

a) **Content-based approach:** This approach needs deep knowledge of your inventory (this is what the existing system tries to achieve - it allocates separate staff to maintain the huge details on every product in the inventory). Using this knowledge each item is profiled based on its similarity characteristics. Thus, a user's tastes are then deduced based on their ratings, purchase history or any other information about their preferences. We use this Content-based approach in our platform with the help of deep learning. The main difference is that there will be no separate staff to enter the details of each product in the proposed system, the deep learning algorithm

understands the abstract details of the image and process the details for recommendations.

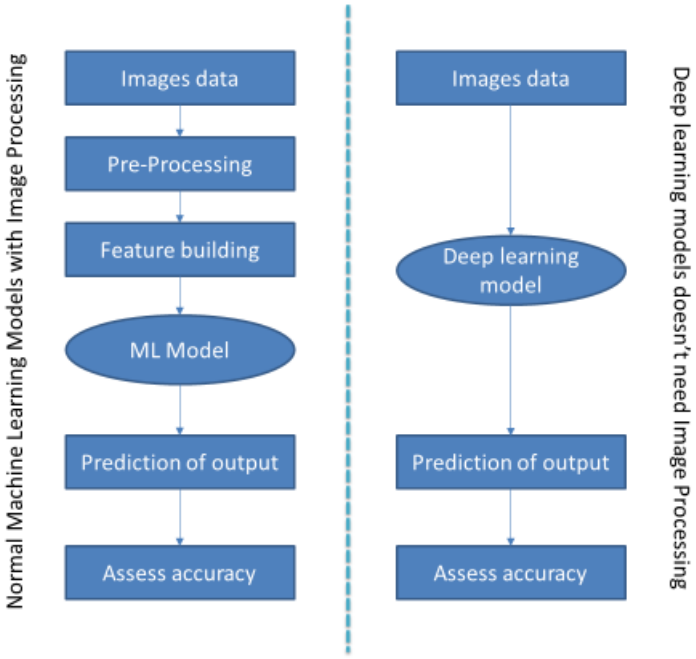
b) **Collaborative filtering approach:** In this approach, every user needs to have an information about their personal tastes. In collaborative filtering approach if a person X has the same opinion as a person Y on any item, then X is more likely to have same opinion as Y on any other item 'A'. In this technique, the process of filtering of information involves collaboration among multiple users, multiple items and so on... etc.

6.4. Computer Vision

Computer vision is a field where computers deal with images or videos to gain understanding from them and extract details from them. Its main purpose is to automate the human visual system to do various things on multimedia data like images and videos. Some tasks include image processing, image analysing, image edges detection, perspective and skew corrections... etc. As all the images are high-dimensional data with numerical values at pixel levels, it needs high computational effort to do these kinds of processing for numerous number of images. In olden days, machine learning models are constructed on the data that comes out of the computer vision process where every abstract detail is extracted from the image and hand-crafted into the data as features to improve the model. In our platform, we automate the automation of feature building using a technique called deep learning.

6.5. Image Processing

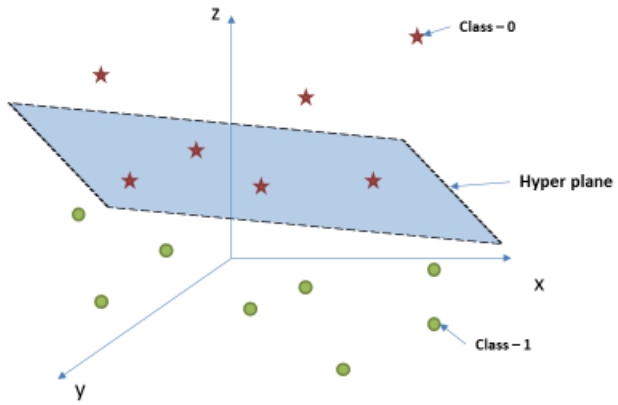
Image processing is another sub domain of computer vision where a digital image is taken as input and processed image will be the output. The processing includes enhancing the image or extract details from the image or remove noise from the image... etc. Usually the image is treated as a two-dimensional array where every point is the pixel data. The process of modelling with the help of computer vision and image processing is explained in detail with the help of a flow diagram below:



Conventional ML Models vs Deep learning

6.6. Support Vector Machines (SVM)

Support Vector Machines are one of the most advanced machine learning algorithm used for regression and classification. Although, it is primarily used to do classification using the concept of hyper planes that define decision boundaries around each class in the data. A hyper plane is a concept that separates multiple set of classes by defining a plane in the hyper space. It constructs (n-1) hyper planes in a multidimensional space that rotates the existing data into a multidimensional space where multiple planes are drawn to separate multiple classes in the data. In this platform, we are planning to do a binary classification (like or not) so a single plane is draw in multidimensional space to separate the data with two classes achieving as much as accuracy as possible. Usually SVMs are mostly used in the case of image classification problems like hand written digit classification. SVM Classification can be represented in the form of a diagram as below:



SVM Classification

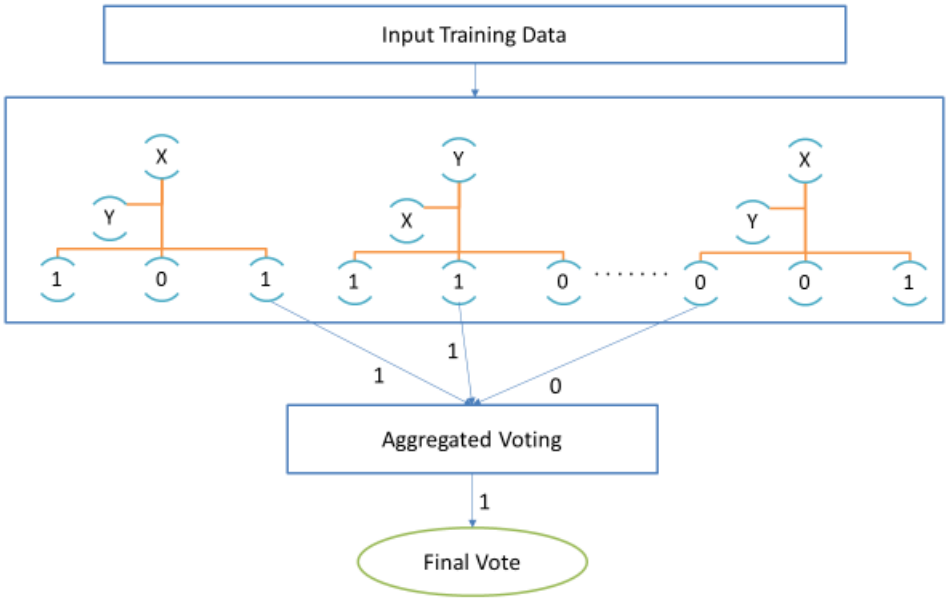
6.7. Random Forest

A Random forest algorithm is a collection or ensemble of multiple decision trees (usually decision stumps with one level) where each tree can vote on a point. In the case of classification, multiple samples of data are passed through multiple trees (i.e. each tree sees different data) and model is constructed at every tree. Thus, when a new data comes every tree in the collection votes for a class and usually the majority voting is considered as the final prediction. This entire concept is called bagging. Bagging is a mix of two words:

a) **Bootstrapping:** A sample of data taken from the training data with replacement is called bootstrap sample. Each tree in the collection gets a bootstrapped sample from the training data which is called an **in-bag** data. The model at each tree is tested on its own left out data called **out-of-bag** data. Thus, the chance of overfitting reduces at a greater extent.

b) **Aggregation:** The final votes from each tree is aggregated and considered as the final prediction for the data point. This is also called as maximum voting algorithm

Thus, in random forests there is a lot of variance in between each tree which reduces variance on the overall model. The bias in the prediction is also very low making the model very stable. Random forest is represented in the form of diagram below:



Random forest Classification

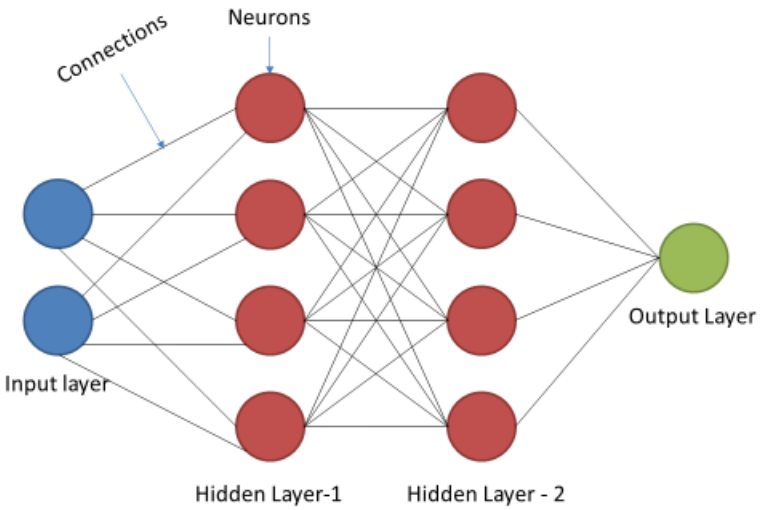
6.8. Artificial Neural Networks (ANN)

ANN is defined by **Dr. Robert Hecht-Nielsen** (inventor of first neuro computers) as "*A computing system made up of several simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.*" In general, ANNs are referred to human brain as their functional behaviour resembles to our brain. Typical neural networks are organized in the form of three layers

1) Input layer
2) Hidden layers
3) Output layers

and each hidden layer will have multiple number of interconnected nodes which contain a function to process the signal from input and this function is called activation function. The process of classification in ANN is achieved in the following way. The input node transmits the data point through the hidden layers where multiple processing steps are done on the point and a final output is achieved at the output layer. Then, the output achieved is compared with the original output and the error is back propagated through the network which causes the adjustment of weights in the hidden layers. i.e. most ANNs

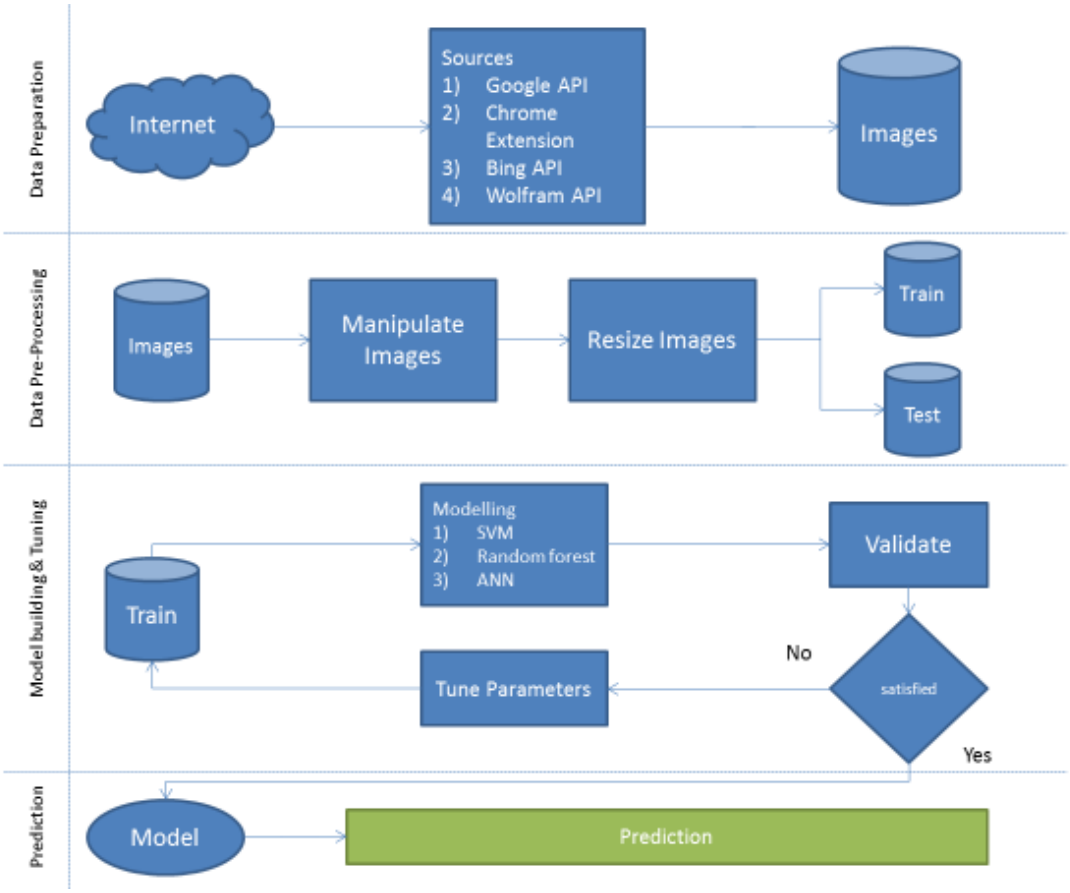
formulate a simple learning rule that modifies weights of the connections based on the error propagated backwards. This process is performed multiple times until a minimized error is achieved. The functioning of a neural network is represented in the below diagram.



ANN Classification

7. System architecture

The System architecture of our platform is given below in the form of diagram.



System Architecture

Our platforms architecture is broken down into 4 important components/modules/layers. They are:

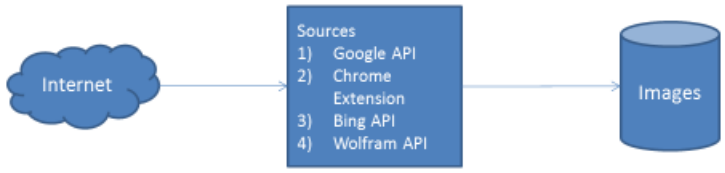
1) Data preparation layer
2) Data pre-processing layer
3) Model building and tuning layer
4) Model prediction layer

Each layer is interconnected to the other one to end up with a full-fledged deep learning taste profile generation engine. The data preparation layer fetches the images of fashion items from the internet. It uses the sources like Google, Bing and Wolfram alpha search engines. In the data pre-processing layer the images data is extracted, processed and

transformed in the required format (i.e. scaling and resizing of each image is performed here). Then any new user logged in is asked to take a survey on a sample of fashion items where he votes every item based on his taste (i.e. he votes whether he likes the shirt or not). This data is stored in My SQL and used as a labelled dataset for performing classification and recommendation. The model building layer uses the labelled data obtained from the pre-processing layer. Multiple machine learning models (like SVM, Random forest, ANN... etc.) are built on top of the data and are evaluated on a validation set. All these models are tuned continuously to achieve a high-level accuracy. Once the model parameters are tuned, the model is rebuilt and again re-evaluated. This process continues until the threshold level of accuracy is achieved. The final Model prediction layer uses the model built from the above layer to perform prediction on new arrivals. The top 'k' items that are having highest ratings for a user are marked as recommendations. These recommendations are continuously tuned based on the user purchase history. Each layer of the architecture is discussed below in detail.

1) Data Preparation Layer

The data preparation layer can be represented as a flow chart as shown below.



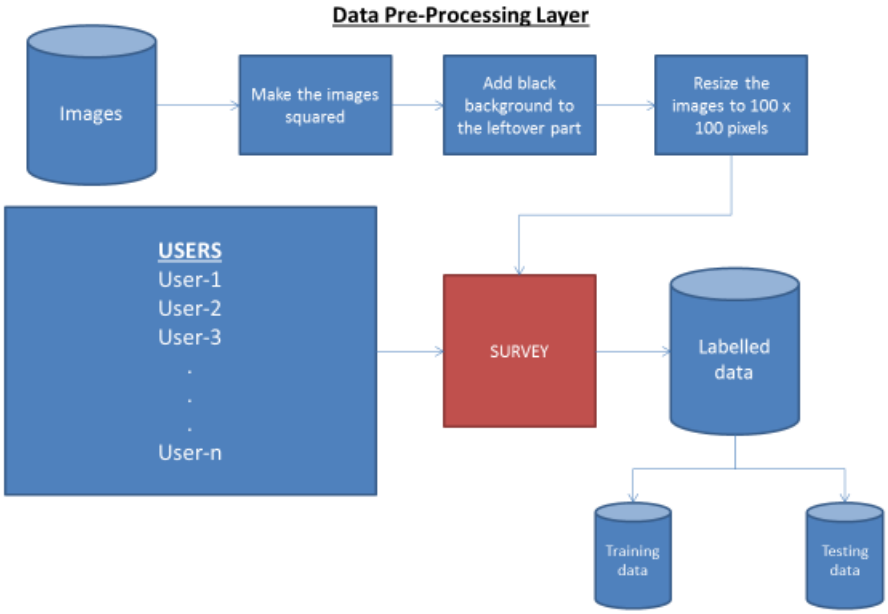
Layer – 1: Data Preparation layer

In this layer the platform fetches the data from the internet using multiple sources. Some of the sources are Google (using python-google API), Bing (using python-Bing API), Chrome extension called ***Fatkun Batch download*** and Wolfram Alpha API. The data we download is images of fashion items particularly Shirts for men. Every image we download is of format of JPEG and each image is different from the other in colour, type and pattern. All these images are downloaded and put in a folder in our local machine. We need to pre-process these images to perform classic machine learning algorithms. But,

when using Deep learning we does not need any pre-processing steps as the features are automatically detected in every image by the model itself.

2) Data Pre-processing Layer

The data pre-processing layer can be represented as a flow chart as shown below.



Layer – 2: Data Pre-processing layer

In this layer, different mechanisms are used to do data pre-processing based on different machine learning algorithms. In case of standard machine learning algorithms like SVM, Random forest, ANN we need to pre-process every image by enhancing the image, scaling the image and resizing the image. But, in case of deep learning algorithms all these steps are not needed because the algorithms itself understands the necessary abstract features and builds them accordingly thus reduces human effort and improves accuracy. Initially we have implemented some of the standard machine learning algorithms on the images data. So, we must do the necessary pre-processing steps as described. As all the

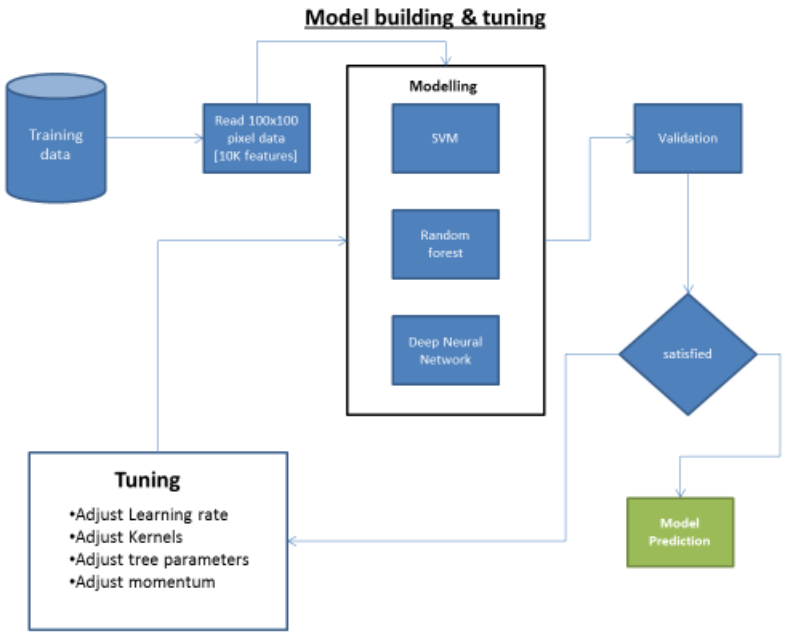
images, will be of different resolutions, we need to standardize them into a single fixed resolution to do that we have done the following steps.

a) Add a black background to every image so that each image becomes a squared image. i.e. if an image is of resolution 300 x 1000 we extend the width from 300 to 1000 pixels by adding black as the background.
b) Once we achieve a square image we then resize it to 100 x 100-pixel resolution for easy processing. Thus, we get all the images into a single resolution (i.e. 100 x 100 pixels).

Note that the above mentioned two steps are not necessary in case of building the deep learning algorithms. The next step of pre-processing is to obtain labels to perform a supervised learning. So, every registered user undergoes a survey where a sample of images are shown and the user needs to deliver his vote on the item whether he likes it or not. Once this process is completed we end up having a labelled data of images which can be used to build supervised models. Then the next step is to divide the labelled data into training and testing. The training data is used to build the model and the testing data is used to validate and tune the model.

3) Model Building and Tuning

The Model building and tuning layer can be represented as a flow chart as shown below.



Layer – 3: Model building and tuning layer

In this layer, the training data with labels for a user (after the survey) is used as input to build the model. As already mentioned each image (Say, we have 1000 images and 700 are used for training) is of size 100 x 100 pixels i.e. each image has 10K features and a binary label classifying whether the user likes it or not. We input this data into multiple machine learning algorithms to perform the task of classification. Each machine learning algorithm uses different mechanisms to do the task of classification. The overview of mechanisms is discussed below and detailed description of these algorithms are in the section of Core concepts.

a) SVM

The data in multi-dimensional (n-dimensional) space is classified by constructing n-1 (in this case 1) hyper planes that gives clear separation of data with different classes (in this case it is binary classification)

b) Random forest

This algorithm builds multiple decision trees where each tree gets different samples of training data and registers their vote on the newly arrived sample. We then consider the aggregation of all these votes to come up with final voting.

c) Artificial Neural network

This builds a network where the input is passed through the input layers and processed through the hidden layers and predicted in the output layer. The error is then back propagated back to modify the weights.

Once all these models are built, we evaluate these models based on three measures:

1) **Accuracy** - It is the accuracy of the model in classifying the likes and dislikes.
2) **F-Score** - It is the weighted trade-off between recall and precision.
3) **Error rate** - It is the error between the actual class and the predicted class.

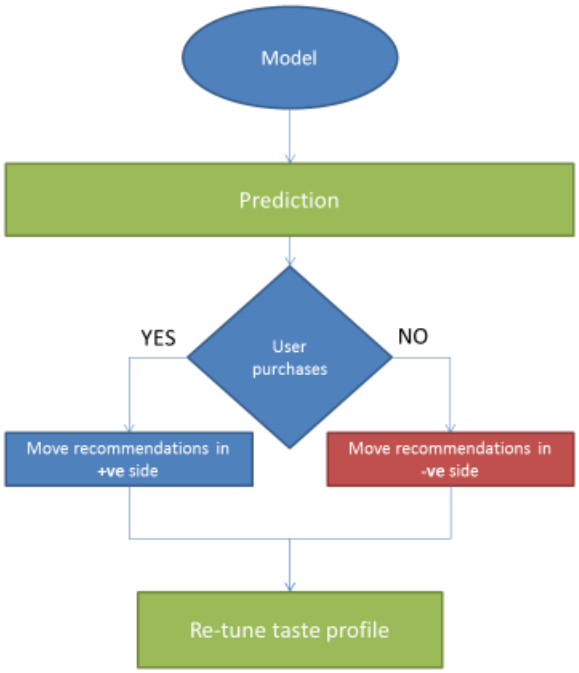
If all these measures are satisfactory, we use the model as the final model for prediction. If these measures are not equalling human level accuracy, then we need to re-tune the models by adjusting the hyper parameters and validate the model again. Some of the hyper parameters for each model are discussed below

- In case of Random forest
  a) ntree = number of trees to be built
  b) mtry = number of variables to be tried at each split
  c) nodesize = size of the variable at every node
- In case of SVM
  a) kernel = kernel to be used for classification. It can be Linear, Spiral, RBF
  b) learning rate = learning rate to approach the gradient descent. The greater the learning rate the faster the model converges. But, the lower the learning rate the accurate the model will be.
- In case of ANN
  a) learning rate = same as above.
  b) hidden = number of hidden layers.
  c) nodes = number of neurons at each layer.

All these parameters are to be tuned continuously to achieve the minimum required accuracy. Once the accuracy is achieved the model object is saved and used in the final layer for prediction.

4) Model Prediction

The Model Prediction layer can be represented as a flow chart as shown below.



Layer – 4: Model building and Tuning

The final model saved from the above layer is used as one of the input in this layer. The other input will be a newly arrived fashion item in the stock. Each item goes through the model and rated between 0 to 1. The top 'K' highest ratings will be sent as recommendations to the user. From this step the tuning of recommendations are done based on user page crawling behaviour. i.e. If the user purchases the recommendation or wish lists the recommendation, then his taste profile swings little bit more towards the positive direction. If the user clicks a dislike option, then his taste profile swings more towards negative direction. Once the direction is known we then re-tune the recommendations by modifying his taste profile. This process is repetitive and never ending.

8. Implementation

This platform is implemented in a layer by layer manner. Each layer has its own implementation and its own code for implementation. We will discuss the methodologies used for implementation as per the System architecture in the following points.

1) Data Preparation layer:

As mentioned in the architecture we download the data from the internet using different sources of connections. For training a standard machine learning algorithm we need around 1000 images where at least 700 of them can be used for training and the other 300 can be used for validation and tuning. To download these images, we have followed various mechanism and used various sources. They are:

We have tried using the Googles API with python and ended up retrieving just 100 results as the API is deprecated currently. The code snippet for the usage of this API is attached below.

```
# # # # # # # # # # # # # # # # #
# # GET IMAGES FROM GOOGLE  # #
# # # # # # # # # # # # # # # # #
import urllib2
import simplejson
import cStringIO

fetcher = urllib2.build_opener()
searchTerm = 't-shirt images'
startIndex = 0
searchUrl = "http://ajax.googleapis.com/ajax/services/search/images?v=1.0&q=" + searchTerm + "&start=" + startIndex
f = fetcher.open(searchUrl)
deserialized_output = simplejson.load(f)

# to access the first image, you need to do this:
imageUrl = deserialized_output['responseData']['results'][0]['unescapedUrl']
file = cStringIO.StringIO(urllib.urlopen(imageUrl).read())
img = Image.open(file)
```

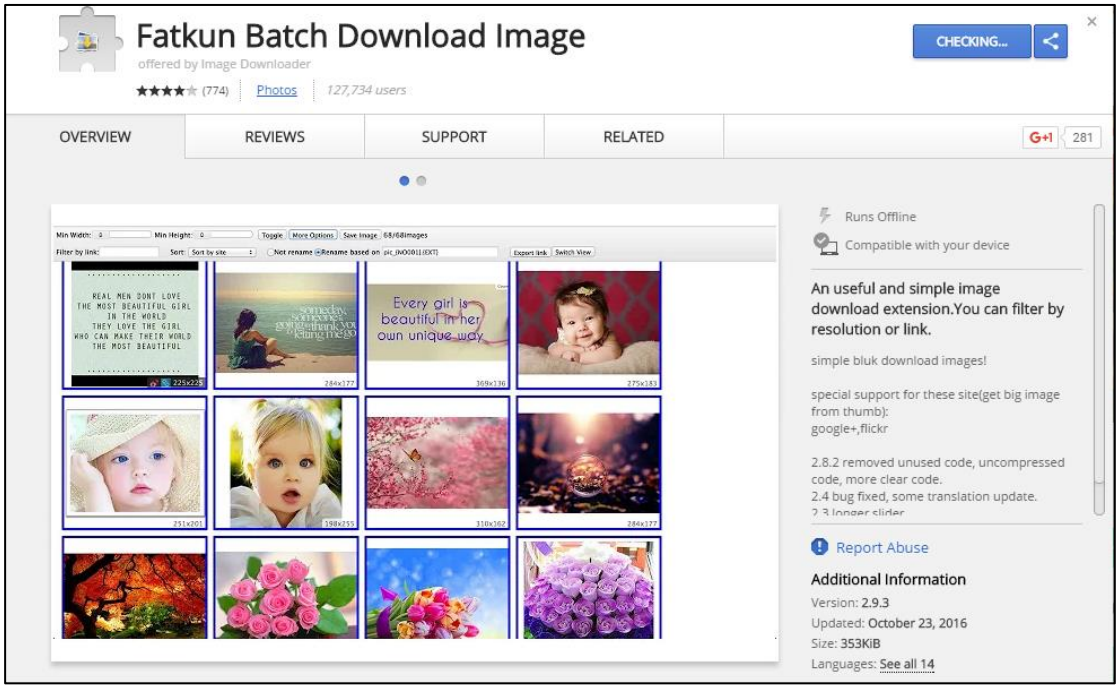
Screenshot – 1: Code snippet for Python-Google API

We have also tried using the Bing-Python API and got few images of around 300. The code snipped for the usage of this API is attached below.

```
# # # # # # # # # # # # # # # # #
# #  GET IMAGES FROM BING   # #
# # # # # # # # # # # # # # # # #
from py_bing_search import PyBingVideoSearch
bing_video = PyBingVideoSearch('[redacted]', "t-shirt images")
first_fifty_result= bing_video.search(limit=50, format='json') #1-50
second_fifty_result= bing_video.search(limit=50, format='json') #51-100
print (second_fifty_result[0].media_url)
```

Screenshot – 2: Code snippet for Python-BING API

We finally found a solution that can scale and retrieve around 600 images within no time. It is an extension provided by Google Chrome named "Fatkun image downloader". It allows the user to download all the selected images in a google search or any page. The screenshot of the extension is attached below.



Screenshot – 3: Fatkun batch download Image – A Chrome extension

Once the images are downloaded they are put together in a folder and renamed using python code for easy access. The naming convention we followed is "shirt_[***NO***].jpg". The code snippet for renaming and screenshot of the folder with images are attached below.

```
import os
import glob

path = "C:\Users\ka294056\Desktop\Kartheek\shirts" # sys.argv[1]

os.chdir(path)

files = glob.glob("*.*")

count = 1
for eachFile in files:
    extension = eachFile.split(".")[-1]
    os.rename(eachFile, "shirt_" + str(count) + "." +  extension)
    count = count + 1
```

Screenshot – 4: Code snippet for renaming images

Screenshot – 5: Shirt images downloaded and renamed

2) Data Pre-processing layer:

As discussed in the architecture, to apply conventional machine learning algorithms on the images we need to do a lot of pre-processing. The main pre-processing steps that needs to be done are:

a) Scale the image to 6000 x 6000 pixels and add black background to the left-over part. This helps in converting a rectangular image to a squared image without cropping.

b) Resize the image to 100 x 100 pixels. This helps us in speeding up the training time.

Once the pre-processing steps are done, we need to save the images in a folder named "Edited"

The code snippet for the above steps and the screenshot of the pre-processed images are attached below.

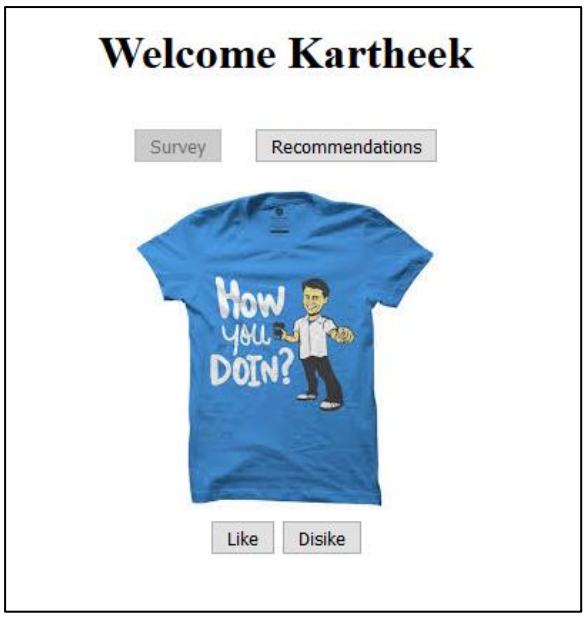
```
import os
from PIL import Image
import glob
###############################################################################################
# This is used to form a black background and scale the image to 6000x6000 nearly
###############################################################################################
def addBlackBG(image, outputPath = os.getcwd()):
    if not os.path.exists(outputPath + "/Edited"):
        os.makedirs(outputPath + "/Edited")
    img = Image.open(image)
    "return a black-background-color image having the img in exact center"
    size = (max(img.size),)*2
    layer = Image.new('RGB', size, (0,0,0))
    layer.paste(img, tuple(map(lambda x:(x[0]-x[1])/2, zip(size, img.size))))
    square_one = layer
    square_one.resize((100, 100), Image.ANTIALIAS)
    square_one.save(outputPath + "/Edited/" + image[:-4] + "_edited.jpg")

###############################################################################################
#################### To resize the Image and make thumbnails ####################
###############################################################################################
def getThumbnails(image, outputPath = os.getcwd(), sizes = [[100, 100]]):
    if not os.path.exists(outputPath + "/Thumbnails"):
        os.makedirs(outputPath + "/Thumbnails")
    for size in sizes:
        img = Image.open(image)
        img.thumbnail(size)
        img.save(outputPath + "/Thumbnails/thumbnail_" + ".".join(image.split(".")[:-1])
```

Screenshot – 6: Code snippet for Scaling and Resizing the Image

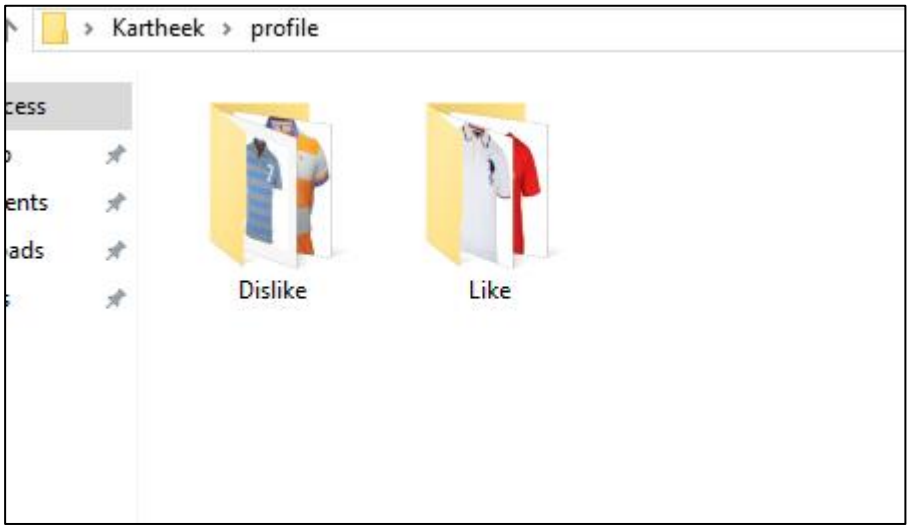
Screenshot – 7: Pre-processed Shirt Images

Once we get the pre-processed images, we need to use the original images to generate training data from a new user. This can be achieved by designing a website using PHP with My SQL. The site has a login where a user logs in and every new user needs to take a survey. The screenshot of the survey is attached below.



Screenshot – 8: Website Survey Page for a User

As per the screenshot the user needs to give his vote on a sample of shirts by clicking one of the two buttons i.e. Like or Dislike. Once the survey ends the data regarding his votes are stored in My SQL and a profile is created for the user that needs to be trained. The screenshot of the profile is attached below.



Screenshot – 9: Profile generated after survey for a User

Out of these images from the profile we take 700 of the balanced samples as training data and the other 300 we use it as testing data for model validation and tuning.

3) Model building and tuning:

As mentioned in the architecture, we use the training data obtained from the above layer (Data pre-processing layer) for training the conventional machine learning models. Each image of size 100 x 100 is taken as the input along with the taste flag (like/not). As all the images are coloured we will have three layers for each image, they are Red, Green and Blue (RGB). Thus, the input becomes a 3-Dimensional array of 100x x 100 pixels each. By vectorising the entire array, we end up with 100 x 100 x 3 = 30,000 attributes and taste flag. So, the entire input training data of 700 images will be a matrix with 30,001 columns and 700 rows. Using this matrix, we have tried applying three standard conventional machine learning algorithms. The snippets and the results are attached below.

i. We have tried fitting a standard SVM model (in R) to the data with kernel setting as Radial Basis kernel (RBF). This RBF is considered as the most standard and optimized kernel that fits better for almost any data. The model is then tested on the test set and achieved an accuracy of around 76% with an f-score of around 0.8. The code snippet and the output is attached below.

```
svmFit = function(trainData, testData) {
    library(e1071)
    fit = svm(Y ~ ., data = trainData)
    prediction = predict(fit, testData)
    return(prediction)
}
```

```
svm[,c(1,2,3,4,9)]
precision    recall    fscore accuracy      rmse
0.7755102 0.8444444 0.8085106     0.76 0.4898979
```

Screenshot – 10: Code snippet for SVM Classification and its Performance

ii. We then tried fitting a very powerful ensemble model called Random Forest with default parameters. The accuracy achieved is around 78% which is quite low for an ensemble. Thus, we tried tuning different parameters by using a grid search mechanism and ended up achieving an accuracy score of around 81% and an f-score of 0.85. The code snippet for random forest before and after tuning along with the output is attached below.

```r
rfFit = function(trainData, testData) {
    library(randomForest)
    fit = randomForest(Y ~ ., data = trainData)
    prediction = predict(fit, testData)
    return(prediction)
}
```

Screenshot – 11: Code snippet for Random forest classifier

```r
fineTuneRF = function(dataset = NULL, mtryRange = 1:15, ntreeRange = seq(1000, 2500, 500), seed = NULL) {
    # # Install required packages if not available
    if("caret" %in% rownames(installed.packages()) == FALSE) {
        install.packages("caret", repos = "http://cran.us.r-project.org/")
    }
    if("randomForest" %in% rownames(installed.packages()) == FALSE) {
        install.packages("randomForest", repos = "http://cran.us.r-project.org/")
    }

    library(caret)

    customRF = list(type = "Classification", library = "randomForest", loop = NULL)
    customRF$parameters = data.frame(parameter = c("mtry", "ntree"), class = rep("numeric", 2), label = c("mtry", "ntree"))
    customRF$grid = function(x, y, len = NULL, search = "grid") {}
    customRF$fit = function(x, y, wts, param, lev, last, weights, classProbs, ...) {
      randomForest(x, y, mtry = param$mtry, ntree=param$ntree, ...)
    }
    customRF$predict = function(modelFit, newdata, preProc = NULL, submodels = NULL)
       predict(modelFit, newdata)
    customRF$prob = function(modelFit, newdata, preProc = NULL, submodels = NULL)
       predict(modelFit, newdata, type = "prob")
    customRF$sort = function(x) x[order(x[,1]),]
    customRF$levels = function(x) x$classes

    control = trainControl(method = "repeatedcv", number = 10, repeats = 3)
    tunegrid = expand.grid(.mtry = mtryRange, .ntree = ntreeRange)
    if(!is.null(seed)) {
        set.seed(seed)
    }
    custom = train(factor(Target) ~ ., data = dataset, method = customRF, metric = "Accuracy",
        tuneGrid = tunegrid, trControl = control)
    plot(custom)
    return(list(fit = custom, summary = summary(custom)))
}
```

Screenshot – 12: Code snippet for tuning Random forest classifier

```
> rf[,c(1,2,3,4,9)]
  precision    recall    fscore  accuracy      rmse
1 0.8078818 0.9111111 0.8563969 0.8166667 0.4281744
```

Screenshot – 13: Performance of Random forest after tuning

iii. We have also tried building a Neural network around the data. The network didn't fit to the data as expected resulting in an accuracy around 65%. Although after tuning the network by changing the hyper-parameters we have ended up with an accuracy score of around 70%. The code snippet and the output for the network is attached below.

```r
# # Build and tune artificial neural nets
annFit = function(trainData, testData) {
    library(neuralnet)
    nnfit = neuralnet(Y ~ ., data = trainData, linear.output = FALSE,
        likelihood = TRUE, err.fct = "ce", hidden = 2)
    prediction = round(compute(nnfit, testData[-length(ncol(testData))])$net.result)
    return(prediction)
}
```

Screenshot – 14: Code snippet for Neural networks classifier

```
tuneANN = function(trainData, testData) {
    library(neuralnet)
    for(alpha in 2:10)
    {
        nHidden = round(nrow(trainData)/(alpha*(3+1)))
        nn = neuralnet(Y ~ ., trainData, linear.output = F, likelihood = T, err.fct = "ce", hidden = nHidden)

        # Calculate Mean Squared Error for Train and Test
        trainMSE = mean((round(nn$net.result[[1]]) - trainData$Y)^2)
        testPred = round(compute(nn,testData[-length(ncol(testData))])$net.result)
        testMSE = mean((testPred - testData$Y)^2)

        print(paste("Train Error: " , round(trainMSE, 4), ", Test Error: ", round(testMSE, 4),
            ", #. Hidden = ", nHidden, sep = ""))
    }
}
```

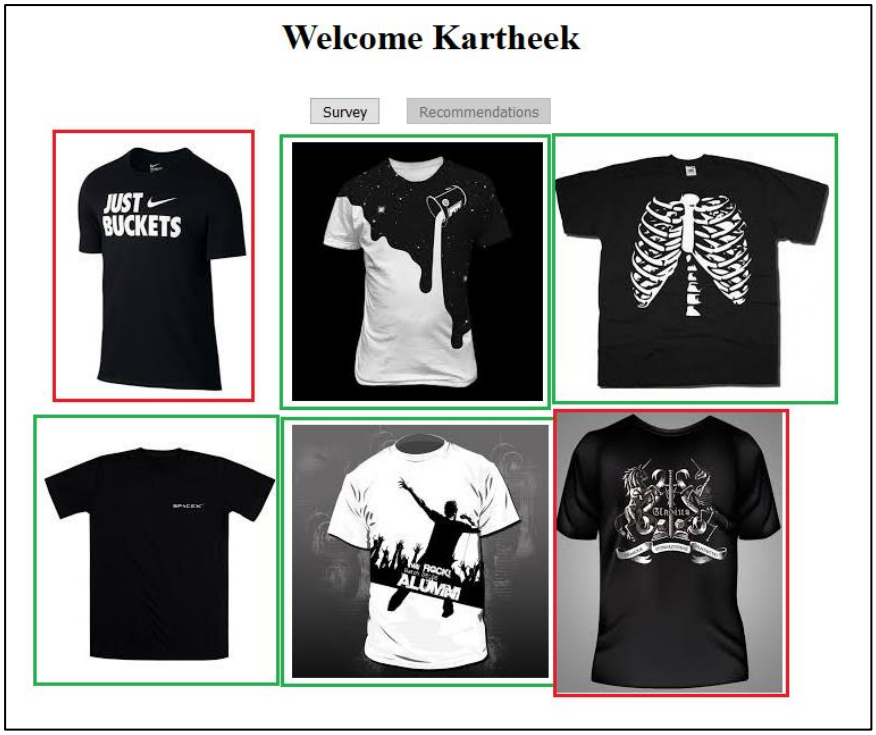
Screenshot – 15: Code snippet for tuning the Neural Networks

```
> ann[,c(1,2,3,4,9)]
  precision    recall    fscore accuracy      rmse
1 0.7368421 0.7777778 0.7567568      0.7 0.5477226
```

Screenshot – 16: Performance of Neural networks after tuning

4) Model Prediction:

The final step in the architecture is model prediction. In this step, we have chosen the random forest model as our final model as it out performed all the others. Using this model, we have predicted the score on 100 new arrival shirt images. The shirts are then sorted based on the score in descending order and the top - 6 (K = 6) are sent as the recommendation for the user. Per the user review on the shirts we have found 4 out of 6 recommended shirts are matching his taste. The screenshot for the recommendations is attached below.



Screenshot – 17: Recommendations for the User

In the above screenshot the shirts marked with green border are the ones that are matching the user taste and the shirts marked with a red border are against the user taste. This concludes that we have achieved an accuracy of 70% which can be boosted by building a deep learning model.

9. Future Work

As discussed in the above sections, we have only implemented the very core concept of the platform. There are many updates to be done in helping the model to improve the visual aspect, and improve the performance of the recommendations. Some of them are mentioned below:

a) **Feature building:** For the conventional machine learning algorithms to work we need to understand the input images better that will help in understanding the patterns for every user. To do this, we need to hand craft few features that may help in the improvement of the model. Some of the planned features are:

- **The dark or light flag:** The centre pixel for each image is extracted and is classified into a binary flag whether it is a dark colour or a light colour. We chose the centre pixel because almost all the images have some background around them that may create noise for the classifier. So, to get the exact colour of the item we need to get the centre pixel assuming the item is placed around the centre in the image.
- **Noise Removal:** Before adding the black background and resizing we need to reduce the unnecessary portions of the image as much as possible. Once this is done we can do the mentioned pre-processing steps. This may result in a spike in the improvement of accuracy.

b) **Go Deep:** As the conventional models are not resulting a performance close to human, we need to build a deep learning model especially Convolution Neural Networks. We can use the open source library Tensor flow to do this. CNNs are most popular for functioning on image data as they have auto feature building technique. At each layer in the CNN some abstract details of the image are extracted which will become more abstract in case of next layer. For example: At first layer the CNN learns detecting edges, at the second layer the CNN learns combination of edges and at the final layer the CNN learns the shape of the shirt. Instead of handcrafting the features, the model itself understands them which makes it even powerful.

c) **Re-tuning recommendations:** Once the recommendations are provided for a user, there is a need of re-tuning based on his purchase history. i.e. If he purchases the recommended item, then his taste profile should move

towards a positive direction and if he didn't purchase we may need to move the taste profile in a negative direction. This should be a never-ending process and the main assumption is that the model evolves continuously.

d) **Inception**: Recently google open sourced its own deep learning model that can be implemented using tensor flow. It is called as Inception network. This network is designed to encounter the ImageNet challenge and was trained on 1,000 classes with over 10 million images. We can use this network just by retraining it on our own class samples. This entire process is called "Transfer Learning". This is the current state-of-art in the field of CNNs. The assumption is that it outperforms the perfectly handcrafted CNN model.

e) **Sort the categories:** We can also use this deep learning classifier to sort the categories of new items arrived based on the history of training items. i.e. The classifier will be able to separate the Shirts and Trousers and each category within the shirts also. This will reduce a lot of human intervention. This will also reduce the time for sorting and effort taken for sorting.

10.References

Weblinks

- http://cs231n.stanford.edu/
- http://karpathy.github.io/2015/10/25/selfie/
- http://recommender-systems.org/
- http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/index.html
- https://www.coursera.org/specializations/recommender-systems
- https://www.coursera.org/learn/neural-networks

Papers

- Online shopping recommendation mechanism and its influence on consumer decisions and behaviours: A causal map approach. Expert Systems with Applications, Lee, K and Kwon, S.
- Amazon.com Recommendations Item-to-Item Collaborative Filtering, Greg Linden, Brent Smith, and Jeremy York.
- Google News Personalization: Scalable Online Collaborative Filtering, Google.
- Convolutional Networks and Applications in Vision, Yann LeCun, Koray Kavukcuoglu and Clement Farabet
- ImageNet Classification with Deep Convolutional Neural Networks, Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, University of Toronto.
- Going Deeper with Convolutions, Google Inc.
- Deep Visual-Semantic Alignments for Generating Image Descriptions, Stanford.

Text books

- Recommender Systems: An Introduction, by Alexander Felfernig, Dietmar Jannach, Gerhard Friedrich, and Markus Zanker
- Building a Recommendation System with R, by Michele Usuelli and Suresh K. Gorakala
- Deep Learning, by Aaron Courville, Ian Goodfellow, and Yoshua Bengio
- Python Machine Learning, by Sebastian Raschka
- Learning from Data - Yaser S. Abu-Mostafa, Malik Magdon-Ismail, Hsuan-Tien Lin C