National Data Science Challenge 2019 - Beginner

Product Category Classification - NYRS

1 Objective

- To build a predictive model for extracting the category of e-commerce products automatically given their images and titles, as illustrated in Fig. 1.

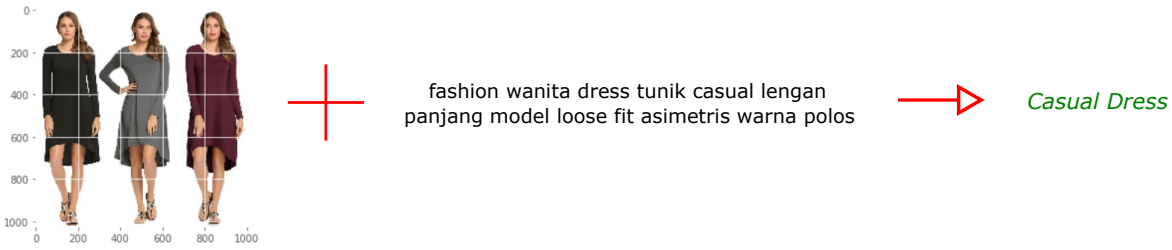


Figure 1: Problem definition: Product category classification.

2 Exploratory Data Analysis (EDA)

We first noted the following aspects of the dataset. There are a total of 58 categories belonging to three top classes: beauty, fashion, and mobile. The number of items within the top categories are unbalanced, with approximately 43%, 33% and 24% of the products belonging to beauty, fashion, and mobile respectively in both train and test set.

The class distribution within the top categories are also unbalanced, e.g. beauty category contains 17 classes, with as high as 28% of the items are classified as powder, whereas only 0.2% are lip-liner. The mobile category has distinct keywords (e.g. iPhone, Samsung, Nokia). In contrast, classification of beauty and fashion products are more challenging as it entail several overlapping words e.g. cream in the most frequent word in both of the 'Other Face Cosmetics' and 'BB & CC Cream' categories. Similarly, lengan, wanita, and dress are common words in fashion.

The items in the beauty and fashion are shuffled in a specific way. For example, top segment of the beauty dataset contains only categories from 0-11, whereas bottom segment constitutes of categories 12-16. This is true for both train and test set.

3 Algorithm

For the current competition, we have adopted an ensembling (stacking) approach, where a gradient boosting (XGBoost) technique is used to combine information from multiple predictive models at the base level (level 0). The model architecture is illustrated in Fig. 2. In general, the stacked model outperforms each of the individual models due to the inherent smoothing nature and ability to incorporate strengths of the base models, while removing their biases. It is thus necessary to have significantly different base predictors for extracting diverse information from the dataset. To this end, we have built three models, each characterizing different aspects of the training set:

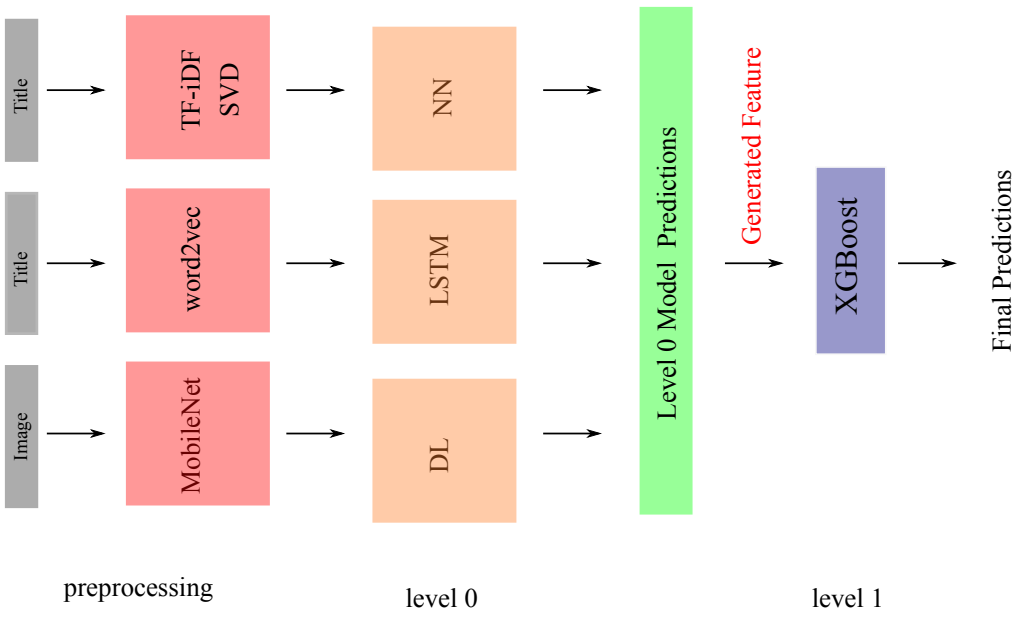


Figure 2: Model Architecture for Product Category Classification by NYRS.

(i) **Image Data**: The Convolutional Neural Networks (CNN) are widely used for image classification. A powerful image classifier can be built by (a) training a CNN from scratch, (b) using the bottleneck features of a pre-trained network (outputs from penultimate layer), (c) fine-tuning the top layers of a pre-trained network. Building a CNN from scratch requires large amount of data. This constraint can be relaxed by adopting the later two approaches, which leverage on an established network pre-trained on a large dataset (e.g. imagenet). Even though fine-tuning, in general, produces better results, it can be computationally expensive. Due to limited computational resources, we thus adopted approach (b). Therewith, each image is first preprocessed and resized to dimension (128,128,3), and subsequently passed through the pre-trained MobileNet. The activations from the penultimate layer (bottleneck features) are forwarded to a Global Average Pooling2D layer and the resulting outputs are stored in memory. We thus encode each image by 1024 dimensional vector. Note that this encoding needs to done only once, thereby reducing computational cost. We chose MobileNet for this dataset, as its performance has been found out to be on par with VGG-16 and InceptionNet v3, while having significantly lesser number of parameters.

A deep neural network is build with two hidden layers, which takes image encoding as input and predicts probability of the image to belong to a particular category. It has been observed that adding Batch Normalization step for each hidden layer gives significant performance boost.

(ii) The title descriptions must be preprocessed into a suitable numerical data structure before feeding into a machine learning algorithm. To achieve this goal, each relevant word in the text corpus can be mapped to a unique point in a finite dimensional vector space. Since the title text is in Bahasa Indonesia, we oped to design a custom word embedding matrix using word2vec model in Gensim. We thus can capture the interactions between words in a meaningful way e.g. the cosign similarity between 'dress' and 'cloth' is higher than 'dress' and 'iPhone'.

The maximum word length of the title feature is determined to be 32. Title word sequences are next mapped into $(32, 300)$ dimensions, with zero padding where necessary. The sequence

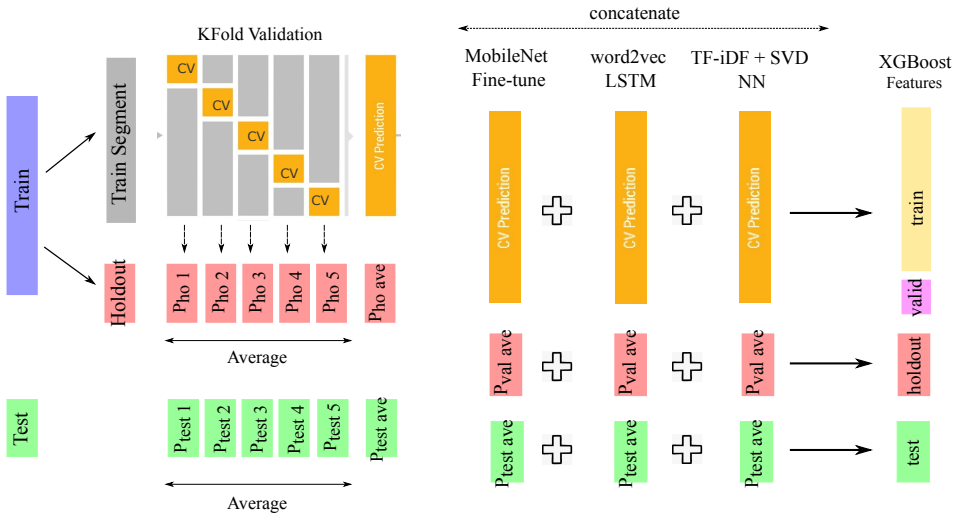
acts as input to a LSTM recurrent neural network. The LSTM based model is found out to be a strong predictor for the product category.

(iii) An alternative way for converting texts to a numerical structure is TF-iDF based vectorization. TF-iDF measures the number of times that words appear in a given title, normalized with respect to its document frequency. It offers diversity into the base predictors, from which the meta learner (XGBoost) can extract additional information. We next performed SDV operation on the TF-iDF matrix to retain useful information using much lesser dimensions. We designed and trained a Neural Network for TF-iDF + SDV based classification.

The class probabilities predicted from the base classifiers are concatenated and supplied as input to the XGBoost model. The XGBoost model helps to build a strong classifier by combining strengths of each base models.

4 Validation

A proper validation strategy is of utmost importance to reconcile the bias-variance problem and determine the expected performance on the test set. Herewith, we have followed the sequence: (i) Create a holdout set comprising of 5% of train data, (ii) make k folds of the remaining train segment, (iii) fit a base model on $k-1$ folds and predict the k^{th} fold, (iv) repeat the previous step for each fold, (v) predict the probabilities untouched holdout and test set with each model trained on $k-1$ folds and take average, (vi) put together out of fold predictions and split it into train and validation set for the XGBoost model, (vii) train the XGBoost model and measure its performance on holdout set, (viii) make the final prediction using XGBoost model.



(a) Base Layer (b) Stacking Layer

Figure 3: The validation strategy for product classification.

5 Discussion

The performance obtained from different models for different top categories are summarized in Table 1. As observed, title descriptions are stronger predictor of product categories than image data.

Table 1: Classification accuracy obtained with predictors for three top categories.

Meta Category	Image	Text LSTM	Text TF-iDF	XGBoost
Beauty	64.02	79.81	75.17	80.60
Fashion	47.75	64.61	59.63	68.44
Mobile	55.28	83.72	80.01	83.70

Notably, the mobile model performs better as the underlying classes have their own unique identifiers. A significant performance gain is achieved in fashion category via XGBoost.

An additional feature is generated (e.g. for fashion) and feed into the XGBoost model to take advantage of the particular shuffling explained in Section 2. For each row in the fashion training and test set (unshuffled), the LSTM model predictions are aggregated to give combined probability (P) of classifying an item within category 17-24. Subsequently, a moving average of P over a window of 5 rows is treated as additional feature, which helps to resolve misclassification between categories $17-24$ and $24-30$, as evident from confusion matrix in Fig. 4.

	Others	Casual Dress	Party Dress	Maxi Dress	A Line Dress
Others	15	15	4	3	0
Casual Dress	5	973	38	57	15
Party Dress	0	57	178	21	2
Maxi Dress	1	151	21	211	4
A Line Dress	1	123	9	19	43
Bodycon Dress	1	169	10	13	7
Wedding Dress	0	2	5	1	0
Big Size Dress	0	40	4	16	2
Tshirt	0	0	0	0	0
Blouse	2	0	0	0	0
Shirt	0	1	0	0	0
Tanktop	0	0	0	0	0
Crop Top	0	0	0	0	0
Big Size Top	0	0	0	0	0

Figure 4: A segment of the confusion matrix on holdout set from the XGBoost Fashion model.

6 Outlook and Acknowledgments

We have identified a few approaches to improve the model performance. For example, it is noted that in the beauty set, the distribution of categories is vastly different in training and test data. Test set contains significantly large number of items from categories 12-16. The oversampling or differential weighing of errors technique could have been implemented. We have developed an automatic image captioning framework (based on train + test data) to provide additional model features. However, due to limited time, we were unable to integrate it in the final model. This approach has potential to capture the interactions between image and title data. Finally, tuning of model hyperparameters can give a significant boost.

We would like to thank Shopee and other partners for conducting such an amazing event. It has been an enriching experience with a steep learning curve. The Kaggle and Medium community provide an ideal atmosphere for learning and practicing data science.