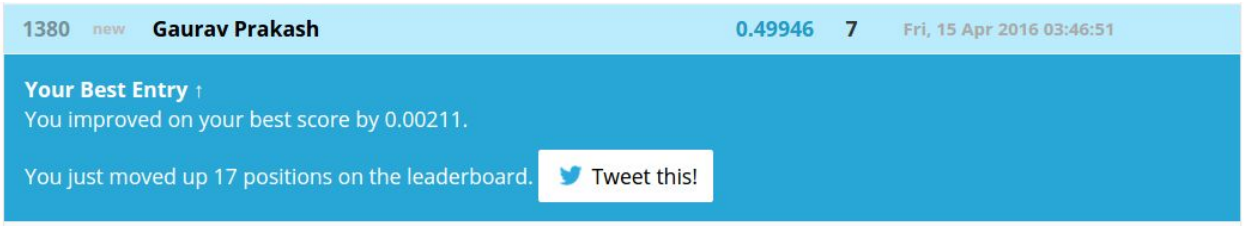
Name: gaurav prakash
mav id: 1001236154

Kaggle username : gprakkash

Kaggle rank and score for the given task : 1380, 0.49946

Screenshot that shows the Kaggle username, rank and score:



Design and implementation details of the program:

Data Processing:

def data_processing(training_data, description_data, attribute_data):
Input:
train.csv, product_descriptions.csv, attributes.csv
Output: merged data

Implementation:

1. The training data, description data, and attributes data is read into pandas dataframe
2. The training data is merged with description data (left join) based on product id
3. The brand name with corresponding product id is extracted from attributes data
4. The brand name is then merged into the result of the previous merge output from step 2 (left join) based on product id.
5. Missing brand names are replaced by empty string
6. All the attributes are merged to form a single string of key value pairs
7. The missing values are replaced by empty string

Feature Extraction:

Input: Output of Data Processing phase (merged data)
Output: extracted features in the form of float values

Implementation:

1. For each record in the merged training data from the data processing stage
2. The search terms is split by space to form 1-grams, 2-grams and 3-grams
3. For each token in n-grams, it is searched in product title, product description, brand name & product attributes. Not just entire tokens in n-grams, but all substring of sizes from 3 to length of token (trimmed from the end of the token) is searched in the above attributes to from features.
4. The features that are built are:
    a. title_count
    b. description_ng1_count
    c. description_ng2_count
    d. description_ng3_count

e. brand_count
f. attributes_count
g. sterms_prop_in_title
h. sterms_prop_in_desc
i. sterms_prop_in_atts
j. sterms_freq_in_brand
k. sterms_matched

Building Random Forest:

1. Each unique relevance value is considered as a class label and all other features are considered as continuous attribute
2. 15 Decision Trees are built for the random forest
3. To build each decision tree, the input is the extracted features from the feature engineering stage
4. Choose an attribute and threshold (binary split) for split:
    a. An attribute is chosen randomly from the set of attributes
    b. All unique values in the attribute are tried as possible threshold for split, to measure Information Gain
    c. The threshold resulting in maximum Information gain is chosen as the best threshold
5. Early stopping:
    a. To avoid overfitting, an early stopping value of 30 is chosen
    b. So while building the decision tree, if at any node there are less than 30 records, it is converted into a leaf node
6. Each leaf node contains the probability of each class

Predicting relevance for Test Records using Random Forest:

For each test record in the test.csv

1. Features are extracted as done for testing data
2. The the feature is send as input to all the 15 decision trees of random forest
3. Each tree output the relevance value (class label) by taking the weighted average of the probability of each class and the value of the corresponding class
4. Thus the relevance prediction from one decision tree is generally not equal to any of the 13 prior unique class labels used while building the tree. This method has helped in improving the rmse by 0.08 as compared to predicting the relevance as the one with max probability for a leaf node
5. The final class prediction is the mean of the relevance values (class prediction) from all the individual 15 trees.

Files:

homedepot.py
documentation.pdf

References:

https://rpubs.com/uvellani/166525