Otto Group Product Classification Challenge

We need to help the company distinguish between their main product categories, given a dataset of products with their corresponding classes. We describe the provided data, preprocessing steps followed by classification using 3 algorithms viz, logistic regression, SVM and random forests. SVM proved to be the best and we submitted the result to Kaggle

1. The Dataset:

Problem: due to our diverse global infrastructure, many identical products get classified differently. Therefore, the quality of our product analysis depends heavily on the ability to accurately cluster similar products.

Objective: to build a predictive model which is able to distinguish between our main product categories i.e., Class_1 to Class_9

Each row corresponds to a single product. There are a total of 93 features for each product, which represent counts of different events. There are nine categories for all products as targets/output/classes. Each target category represents one of our most important product categories from Class_1 to Class_9.

2. Description of methods:

a) **Logistic Regression** - Since we have a multi-label classification, the first choice was logistic regression because it is comparatively quicker and simple to implement and has few parameters to optimize. Also it assumes a smooth linear boundary of classification which is practically pretty intuitive.

b) Random Forrest:

"The Random Forests algorithm is one of the best among classification algorithms - able to classify large amounts of data with accuracy" - Rose Data Science Professional Practice Group

It is an ensemble learning method which involves constructing a number of decision trees at training time and outputs the class that is the mode of the classes output by individual trees or a range of probabilities as required. Each tree predicts independently of the others and then combines all these 'weak' learners into a 'strong' one (ensemble). This overcomes the high variance of single decision trees. We had to tune 3 parameters for optimal performance

c) Support Vector Machines

SVM is a supervised machine learning algorithm which does some extremely complex data transformations, then figures out how to separate data based on the labels or outputs you've defined. It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs. The benefit is that you can capture much more complex relationships between your datapoints without having to perform difficult transformations on your own. The downside is that the training time is much longer as it's much more computationally intensive

3. Experiments

A. Feature selection Analysis: *feature_select.xlsx*

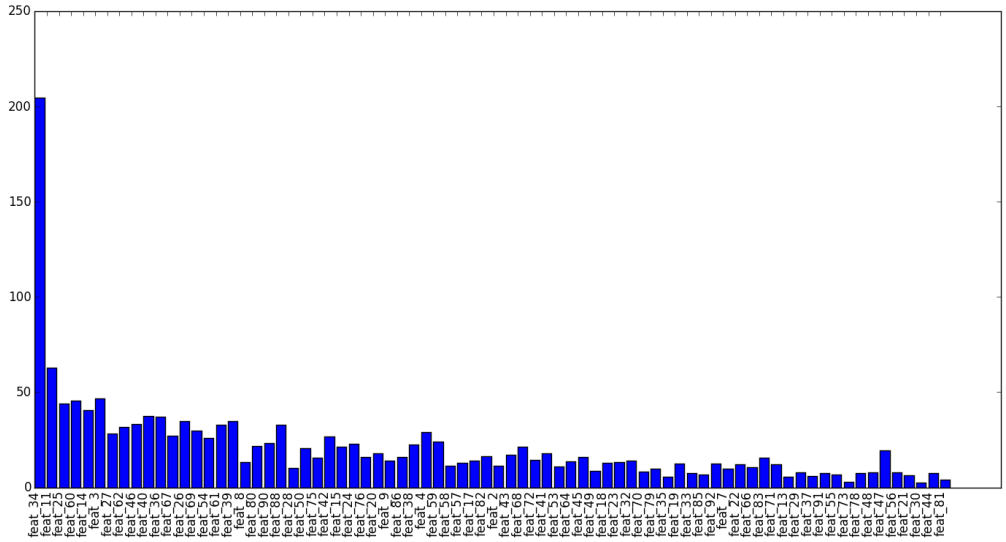
Tab 1: feature prioritization

- Plotted graphs showing amount of effect each feature has on the output
- Did this 10 times and averaged out results
- Arranged features in the decreasing order of importance

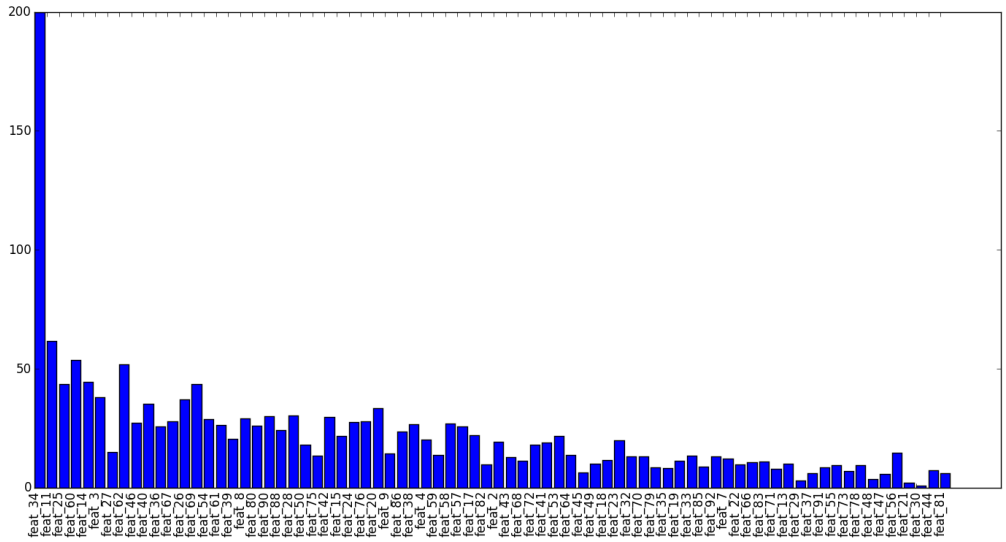
Tab 2: Checking accuracy for a subset of the best features selecting 93-i each time (where i goes from 0 to 70)

- Observed that If we select less than 57 features, accuracy starts to decline sharply
- Ran it 8 times and observed that in general, selecting top 73 to 77 features' giving the best result and our best bet after looking at the data is selecting the best 76 features

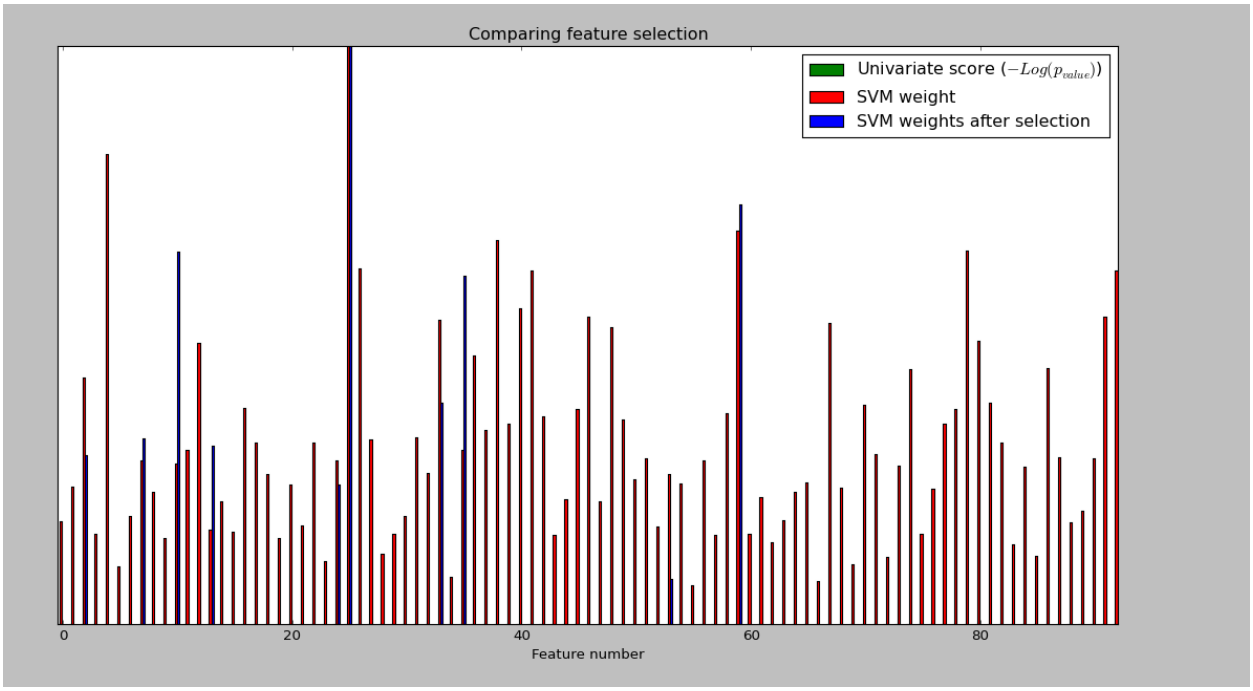
Testing features on Logistic



Testing features on Random Forests:



Testing features on SVM:



Feature selection - SVM

a. We already have selected 76 features as per our previous experiments. We'll do some more analysis and prove these features are correct by testing on svm also.

b. Firstly, we calculate accuracy on complete feature range

10 features, accuracy = 0.56027655298
15 features, accuracy = 0.626452706419
20 features, accuracy = 0.641644943879
25 features, accuracy = 0.657139371154
30 features, accuracy = 0.667962078183
35 features, accuracy = 0.678422520348
40 features, accuracy = 0.685801196745
45 features, accuracy = 0.69886761522
50 features, accuracy = 0.696458959187
55 features, accuracy = 0.703522425126
60 features, accuracy = 0.696845453042
65 features, accuracy = 0.69848952204
70 features, accuracy = 0.697345079637
75 features, accuracy = 0.698390206046
80 features, accuracy = 0.692339669079
85 features, accuracy = 0.697808098581

This clearly suggests that minimum 45% of features need to be selected

Analysis to show features selected should be near 76

1 features, accuracy = 0.295070154688
46 features, accuracy = 0.718791332684
76 features, accuracy = 0.721464747992
93 features, accuracy = 0.700648131856

An analysis on close range features near 76

72 features, accuracy = 0.704291158058
74 features, accuracy = 0.707232586852
76 features, accuracy = 0.702448932571
78 features, accuracy = 0.694436312144
80 features, accuracy = 0.694874968764

B. Parameter optimizations done on a randomly chosen subset of the training set

- Logistic: *logistic_folds_parameter_optimization.xlsx*

Parameter Folds: Ran logistic 9 times for different number of folds (10,15,20), observed than 20 folds is giving a the best average result of 65.98819109 for a small subset of the training set

- Random Forests: *random_forests_parameter_optimization.xlsx*

RandomForestClassifier has 4 parameters:

Ran the algorithm for folds = [10,15,20], n_estimators = [100,150,200], min_samples_split = 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 and min_samples_leaf = [1,2,3,4,5] i.e., ran 540 combinations 4 times (3*3*12*5)

Tabulated summary of the analysis (detailed in the excel sheet *random_forests_parameter_optimization.xlsx*)

Parameter	**Value**	**Avg Accuracy**	**Selected Value**
Folds	10	67.71904762	20
	15	67.7368099	
	20	67.93747987	
n_estimators	100	67.61373792	200
	150	68.03382649	
	200	68.1648752	
samples_per_leaf	1	68.76007326	1
	2	67.95757021	
	3	67.56654457	
	4	67.12179487	
	5	66.53998779	
samples_per_split	5	68.48888889	**6**
	6	68.57777778	
	7	68.35603865	
	8	68.00048309	
	9	68.33381643	
	10	68.19903382	
	11	67.86618357	
	12	67.59806763	
	13	68.13236715	
	14	68.13333333	
	15	67.53285024	
	16	67.75410628	

- SVM analysis:

First, we start with calculating accuracy for different folds of sample training dataset and the mean accuracy for corresponding folds is as follows -

1) Accuracy for folds – [10, 15, 20, 25]

FOLDS	10	15	20	25
MEAN ACCURACY	70.44024	70.69629	70.54995	70.90567

2) Kernel Comparision

linear kernel, accuracy = 75.874994140413
poly kernel, accuracy = 74.4664070035
rbf kernel, accuracy = 75.5457116267
sigmoid kernel, accuracy = 26.0501627458

3) Comparision on different C value and gamma values

C and gamma are meta-parameters used to tune SVM model.
C value - The C parameter trades off misclassification of training examples against simplicity of th e decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly by giving the model freedom to select more samples as support ve ctors.

Gamma parameter - Intuitively, the gamma parameter defines how far the influence of a single t raining example reaches, with low values meaning 'far' and high values meaning 'close'. The gam ma parameters can be seen as the inverse of the radius of influence of samples selected by the m odel as support vectors.

0.01 C, 0.1 gamma, accuracy = 0.71934734605
0.01 C, 1 gamma, accuracy = 0.71934734605
0.01 C, 10.0 gamma, accuracy = 0.71934734605
1 C, 0.1 gamma, accuracy = 0.712178683502
1 C, 1 gamma, accuracy = 0.712178683502
1 C, 10.0 gamma, accuracy = 0.712178683502

4) Gridsearch Result for tuning hyperparameters

Best parameters: {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
The above parameters were the best parameters evaluated after passing the following grid parameters to Gridsearch

param_grid=[{'kernel': ['linear'], 'C': [1, 10, 100, 1000], 'gamma': [0.1, 0.01, 0.001, 0.0001]}, {'kernel': ['rbf'], 'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001]}],

GridSearchCV(cv=5, error_score='raise', estimator=SVC(C=1, cache_size=200, class_weight=None, coef0=0.0, decision_function_shape=None, degree=3, gamma='auto', kernel='rbf', max_iter=-1, probability=False, random_state=None, shrinking=True,tol=0.001, verbose=False),

After getting the best features in Gridsearch method, we executed a 20-fold CV on the entire dataset and below were the observed accuracies

svm.SVC(C=100, gamma=0.0001, kernel='rbf')

[0.78082634, 0.77695287, 0.78624475, 0.7701001, 0.77802908, 0.77479806, 0.77156704, 0.78739903, 0.77447496, 0.77407886, 0.77957337, 0.77820886, 0.78014872, 0.77012609, 0.78499838, 0.77515367, 0.77030087, 0.77903591, 0.77256551, 0.77177083]

4. Conclusion:

- Comparison of algorithms:

We ran a Welch t-test to compare them and got a statistically significant result that suggests one algorithm is better than the other. It also followed the observation of the accuracies and mean accuracy of the 2 algorithms

1. Logistic vs Random forests: Random forests is the better of the two algorithms
2. SVM vs random forests: SVM is the better of the two algorithms

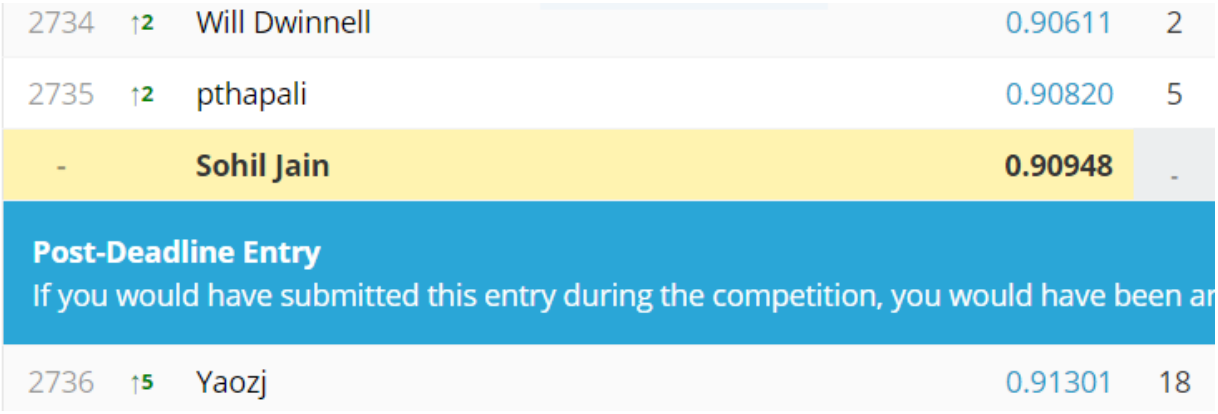
- Future Steps:

We can use multiple algorithms and use them in synergy to give better results

We can also use multi-level approach for instance using adaboost

As far as the dataset is concerned, we might want to look into strategies to visualize it better and also try to remove more features that affect the output negatively. We might want to group the dataset into unique sets with respect to target values

Kaggle submission: We finalized SVM to use in our submission and got a rank of 2735 and was the benchmark



5. Team Work:

Tasks	Logistic, RF	SVM
Analyzing the dataset	Rohit, Sohil	
Selecting the classification models	Rohit	Sohil
Feature selection	Rohit	Sohil
Parameter Optimization	Rohit	Sohil
Algorithm Comparision	Rohit, Sohil	
Documentation	Rohit, Sohil	