Document Classification using Naïve Bayes, Random forest, Linear and non-linear SVM

Table of contents:

* Introduction
* Data Profiling
* Data Cleaning
* Data Pre-Processing
* Feature Selection
* Learning and Applying different Models
* Conclusions

Introduction:

The data-set simulated the real-life scenario of jobs posted on a job portal and comprised of different columns in which most important was Job's title, Job's description along with its category(industry). As the data was labelled so in the context of machine learning, it was a Supervised Machine learning problem i.e. I had access to the data that was already correctly labelled and I had to train a model using this historical data. The main goal was to build a model that could accurately classify new and unseen data when it was input to it i.e. to assign proper class/label/category to a job posting when its input to the model. As the nature of the data was "text" so this project also involved extensive usage of text mining techniques as well. Text in its basic form is unstructured and to develop predictive models, the data needs to be thoroughly pre-processed. So, the pipeline of developing models that I followed was: Data Profiling -> Data Cleansing -> Exploratory Analysis -> Data Pre-processing -> Feature Extraction and Selection -> Model Development -> Model Evaluation.

For model development, I used and compared the following set of machine learning algorithms:

1. Bernoulli Naive Bayes
2. Multinomial Naive Bayes
3. Random Forests
4. Linear SVM
5. SVM with non-linear kernel

and compared these algorithms on different metrics like accuracy, F1-Score, training and testing time. As per my analysis, Random Forest outshines all of the other models when it comes to accuracy. SVM accuracy score was also quite similar although accuracy was lesser than random forest by 0.02% but it outshines all other models when prediction time was considered. For implementation, I used Python. Specifically, I used the following libraries/modules of Python for different set of tasks: pandas, NumPy, sklearn, nltk, matplotlib.

**Applying Naive Bayes:**

Naive Bayes is one of the most widely used classification algorithm in text mining applications. Based on Bayes theorem, this model makes the assumption that all the features are independent of each other and uses the probabilities of each attribute belonging to each class to make a prediction. The condition of independence may not be valid in many circumstances but as a base line model, it’s a good starting point to test its performance on the provided data. There are two forms of Naive Bayes:

1. Bernoulli (designed for Boolean/binary features i.e. just considers the presence or absence of a feature)
2. Multinomial (which also considers the occurrence counts of the feature

### Applying Multinomial Naive Bayes:

Bernoulli Naive Bayes just uses the fact that whether a feature is present or not. However, if we somehow also take into account the occurrence weight or count of the feature as well (in our case, the TFIDF weight of each feature), we can hypothesize that the performance of such classifier will be equally good, if not better.

### Applying Random Forest Classifier:

Random Forests belong to the ensemble learning methods for classification and regression tasks. In Random Forest, a subset of the training data is fit on a number of decision trees. Random Forests have the characteristic to minimize variance if it’s there in the data-set. Their training time is generally quite higher which is one of their drawbacks to be used in production environment. However, the operations can be parallelized to reduce the training time. Also, greater the number of trees in the random forests, better will be the result however it comes with a trade-off: the training time tends to increase as well. Also, if the number of trees tend to increase, the rate of improvement of accuracy may also tend to slow as well. I've used random forests with 50 trees.

### Applying Linear Classifier (SVM) using Stochastic Gradient Descent:

Stochastic Gradient Descent (SGD) is a one of the most efficient approaches used in linear classifiers under convex loss functions such as (linear) Support Vector Machines. It has proven to perform well in in large-scale and sparse machine learning problems.

### Applying SVM with non-linear kernel

SVMs can also be used perform non-linear partitioning. This is achieved by means of a kernel function. In this particular case, I use RBF (Radial Basis Function) kernel and have set a high c value (penalty parameter for error term).

## Conclusion:

So it can be gathered clearly from the metrics that Linear SVM is clearly the winner in terms of training time and testing time, although Random Forest classifiers' accuracy score is also slightly more but its training and prediction time is very high compared to SVM. It has also been found and demonstrated that if we use lesser number of features, selected by applying chi-square test, then the accuracy scores remains the same. If we further analyse the training and prediction time of the machine learning algorithms, we will come to further know that it would be greatly improved as well because of much less number of features. The results have been again validated by making use of cross validation. We saw that the accuracy scores that we got were equal to the ones we got by applying cross validation.