Machine Learning

Table of Contents

1. Problem Identification

Based on the dataset, the requirement suggests that this is supervised text classification problem, and the goal is to investigate which supervised machine learning methods are best suited to solve this.

Given a product description of a particular product, the product category needs to be predicted from one of the 5 categories. The classifier makes the assumption that each new product description is assigned to one and only one category. This is multi-class text classification problem.

2. Data Exploration

Before applying machine learning models, explore some of the product descriptions and the number of products in each class.

For this project, we are considering two columns – “description” and “product_category”

- **Input**: description

Example: `'JewelsWonder Wood Necklace - Buy JewelsWonder Wood Necklace only for Rs. 395 from Flipkart.com. Only Genuine Products. 30 Day Replacement Guarantee. Free Shipping. Cash On Delivery!'`

- **Output**: product_category

Example: `"Jewellery"`

Also, adding a column encoding for product category as an integer because categorical variables are better represented by integers than strings.

```
# View data in dataset
flipkart_df.head()
```

Below is the sample data after factorizing the "product_category" to "category_id" in integer format.

```
df['category_id'] = df['product_category'].factorize()[0]
```

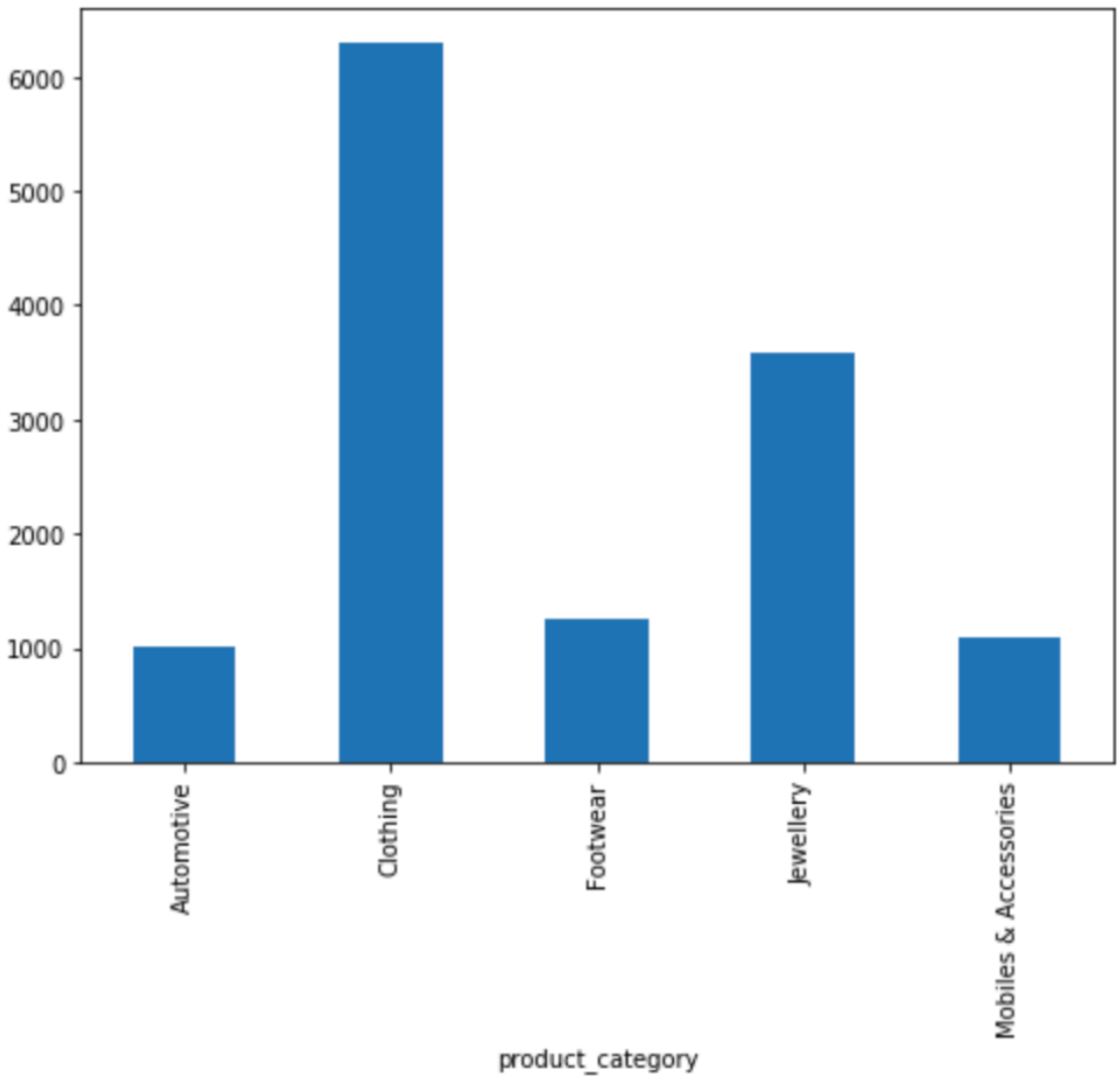
Here, the product category "Automotive" has been mapped to category_id as integer 0.

product_category	description	category_id
Automotive	Specifications of JVC CS J410X Coaxial Car Spe...	0
Automotive	Buy Allure Auto CM 732 Car Mat Ford Endeavour ...	0
Automotive	Buy Allure Auto CM 1180 Car Mat Hyundai Getz f...	0
Automotive	3a Autocare Etios Car Mat Toyota Etios (Beige)...	0
Automotive	Buy Allure Auto CM 2076 Car Mat Maruti Vitara ...	0

3. Imbalanced Classes

The number of products per category are imbalanced. Product descriptions are more biased towards Clothing and Jewellery categories.

```
# Verify if any Imbalanced classes
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,6))
df.groupby('product_category').description.count().plot.bar(ylim=0)
plt.show()
```



In the above case of imbalanced data, the majority classes might be of great interest and is desirable to have a classifier that gives high prediction accuracy over the majority class and maintaining a reasonable accuracy for minority classes.

4. Text Representation

The classifiers and learning algorithms cannot directly process the text documents in their original form, as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length. Therefore, during the preprocessing step, the texts are converted to a more manageable representation.

Use the bag of words model - where for each product description the presence of words is taken into consideration, but the order in which they occur is ignored.

Specifically, for each term in our dataset, we will calculate a measure called Term Frequency, Inverse Document Frequency, abbreviated to tf-idf. We will use `sklearn.feature_extraction.text.TfidfVectorizer` to calculate a tf-idf vector for each of product descriptions.

- **sublinear_df** is set to True to use a logarithmic form for frequency.
- **min_df** is the minimum numbers of documents a word must be present in to be kept.
- **norm** is set to l2, to ensure all our feature vectors have a euclidian norm of 1.
- **ngram_range** is set to (1, 2) to indicate that we want to consider both unigrams and bigrams.
- **stop_words** is set to "english" to remove all common pronouns ("a", "the", ...) to reduce the number of noisy features.

```
# Text Representation: TfidfVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5, norm='l2',
encoding='latin-1', ngram_range=(1, 2), stop_words='english')

features = tfidf.fit_transform(df.description).toarray()
labels = df.category_id
features.shape
```

5. Multi-Class Classifier

- To train supervised classifiers, we first transformed the “Consumer complaint narrative” into a vector of numbers. We explored vector representations such as TF-IDF weighted vectors.
- After having this vector representations of the text, train supervised classifiers to train unseen “description” and predict the “product_category” on which they fall.

After all the above data transformation, now that all the features and labels are available, it is time to train the classifiers. There are a number of algorithms we can use for this type of problem.

Naïve Bayes Classifier: Most suitable for word counts is multinomial variant.

```
# Naive Bayes Classifier: multinomial variant
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
```

```
X_train, X_test, y_train, y_test = train_test_split(df['description'],
df['product_category'], random_state = 0)

count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)

#Fit the training set
clf = MultinomialNB().fit(X_train_tfidf, y_train)
```

After fitting the model to the training set, below depicts the sample predictions

```
# Sample Prediction
print(clf.predict(count_vect.transform(['JewelsWonder Wood Necklace -
Buy JewelsWonder Wood Necklace only for Rs. 395 from Flipkart.com. Only
Genuine Products. 30 Day Replacement Guarantee. Free Shipping. Cash On
Delivery!'])))
```

Predicted category: `['Jewellery']`

	product_category	description	category_id
11456	Jewellery	JewelsWonder Wood Necklace - Buy JewelsWonder ...	3

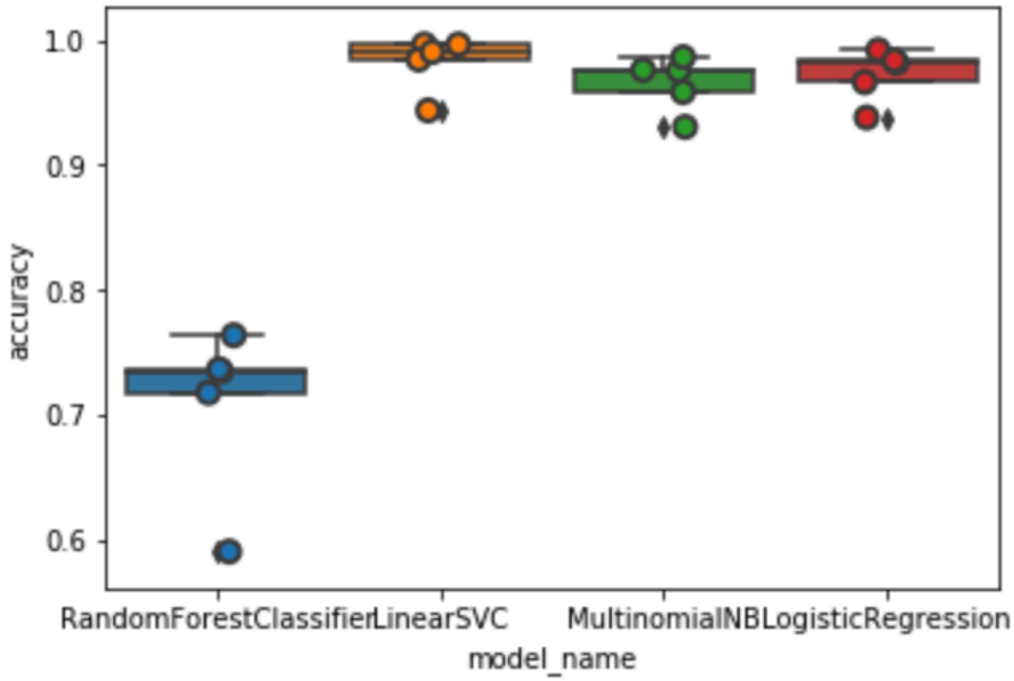
6. Model Selection

Experiment different machine learning models and evaluate their accuracy to find any potential issues.

Consider the following four models:

- Logistic Regression
- Naive Bayes (Multinomial)
- Linear Support Vector Machine
- Random Forest

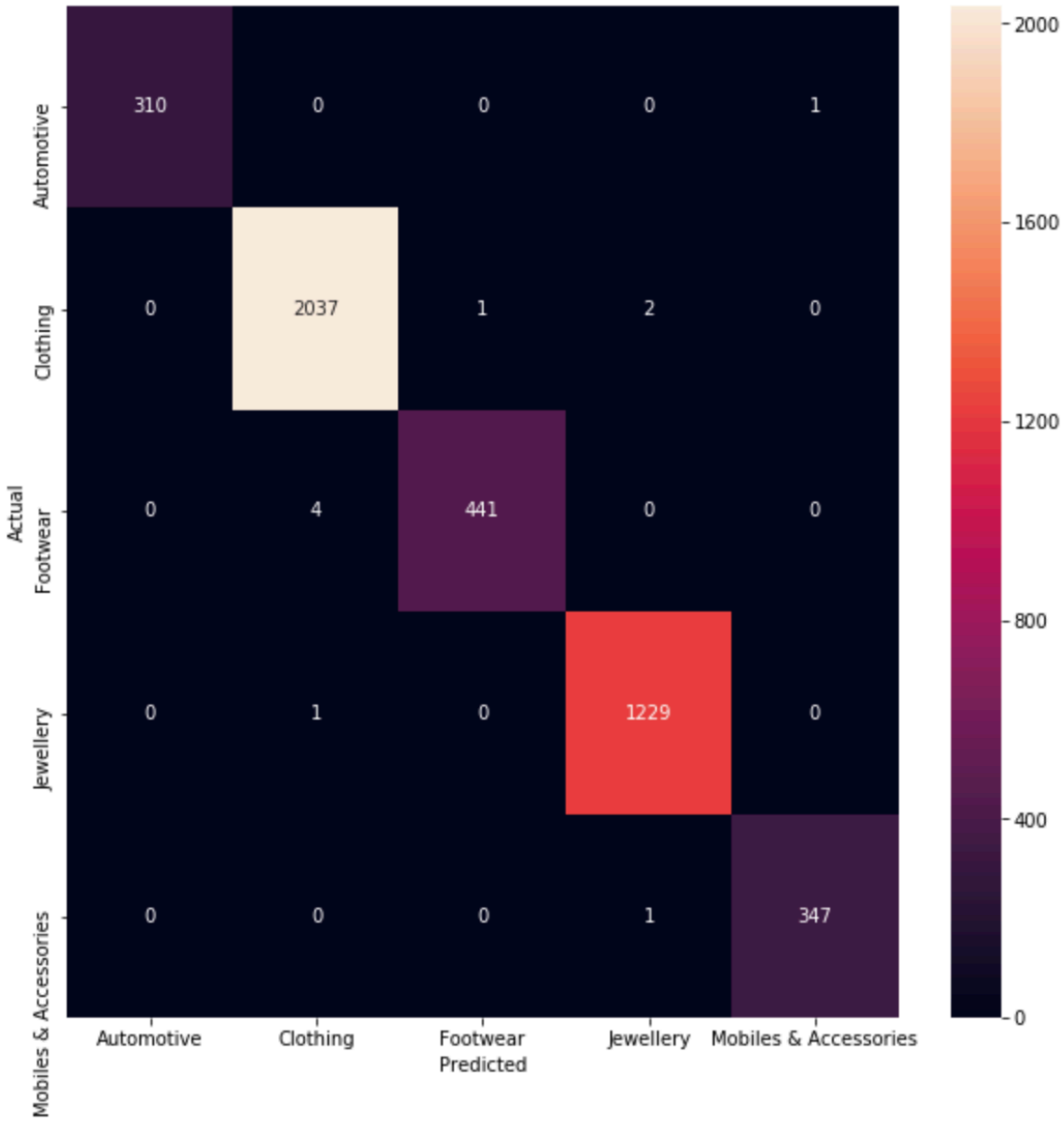
```
# Evaluate accuracy and find the source of potential issues
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.model_selection import cross_val_score
models = [
    RandomForestClassifier(n_estimators=200, max_depth=3, random_state=0),
    LinearSVC(),
    MultinomialNB(),
    LogisticRegression(random_state=0),
]
# 5 - fold cross validation
CV = 5
cv_df = pd.DataFrame(index=range(CV * len(models)))
entries = []
for model in models:
  model_name = model.__class__.__name__
  accuracies = cross_val_score(model, features, labels, scoring='accuracy', cv=CV)
  for fold_idx, accuracy in enumerate(accuracies):
    entries.append((model_name, fold_idx, accuracy))
cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx', 'accuracy'])
```



LinearSVC, Logistic Regression and Multinomial Naive Bayes perform better than RandomForest classifier, with LinearSVC having a slight advantage with a median accuracy of around 98%.

7. Model Evaluation

Considering the highest accuracy model Linear SVC based on above metrics– plot a confusion matrix to find discrepancies between the actual and the predicted labels.



From the above confusion matrix, huge number of predictions are on the diagonal, where the predicted label is equal to actual label, however there are some misclassifications.

For instance, from the following - actual labels are “clothing” but are predicted as “jewellery”

```
'Clothing' predicted as 'Jewellery' : 2 examples.
```

	product_category	description
6114	Clothing	Key Features of Pout Brass Bangle Beautiful Ba...
6038	Clothing	Key Features of classyworld Brass Cufflink Ide...

8. Classification Report

Calculate and print the classification report for each class:

```
# Print out the classification report for each class
from sklearn import metrics

print(metrics.classification_report(y_test, y_pred,
target_names=df['product_category'].unique()))
```

9. Productionizing models

Set infrastructure for the trained model to be used for predictions for other scenarios.