# NLP Text Classification Project

This Jupyter notebook walks through an end-to-end pipeline for multi-class text classification using TF–IDF and Logistic Regression.

## 1. Introduction

**Objective:** Build a model to classify product descriptions into 15 categories.

**Datasets:**

* train\_data.csv: (id, text)
* train\_label.csv: (id, label)
* test\_data.csv: (id, text)

## 2. Setup & Imports

# Data manipulation  
import pandas as pd  
import re  
  
# Text processing  
from sklearn.feature\_extraction.text import ENGLISH\_STOP\_WORDS, CountVectorizer, TfidfVectorizer  
  
# Model & evaluation  
from sklearn.model\_selection import train\_test\_split, GridSearchCV  
from sklearn.pipeline import Pipeline  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import classification\_report, confusion\_matrix  
import matplotlib.pyplot as plt

## 3. Data Loading & Merge

# Load  
train\_data = pd.read\_csv('train\_data.csv')  
train\_labels = pd.read\_csv('train\_label.csv')  
  
# Merge on id  
train = train\_data.merge(train\_labels, on='id')  
train.head()

## 4. Text Preprocessing

def preprocess\_text(text):  
 text = text.lower()  
 text = re.sub(r'<.\*?>', '', text)  
 text = re.sub(r'[^a-z\s]', '', text)  
 tokens = [w for w in text.split()  
 if w not in ENGLISH\_STOP\_WORDS and len(w) > 2]  
 return ' '.join(tokens)  
  
train['clean\_text'] = train['text'].apply(preprocess\_text)  
train[['text','clean\_text']].head()

## 5. Exploratory Data Analysis

### 5.1 Label Distribution

plt.figure(figsize=(6,4))  
train['label'].value\_counts().plot(kind='bar')  
plt.title('Label Distribution')  
plt.show()

### 5.2 Top‑20 Frequent Words

cv = CountVectorizer(max\_features=20)  
freqs = cv.fit\_transform(train['clean\_text']).toarray().sum(axis=0)  
freq\_df = pd.DataFrame({'word': cv.get\_feature\_names\_out(), 'freq': freqs})  
freq\_df.sort\_values('freq', ascending=False)

## 6. Train/Validation Split

X = train['clean\_text']  
y = train['label']  
X\_train, X\_val, y\_train, y\_val = train\_test\_split(  
 X, y, test\_size=0.2, stratify=y, random\_state=42)

## 7. Baseline Model: TF–IDF + LogisticRegression

pipeline = Pipeline([  
 ('tfidf', TfidfVectorizer(max\_features=1000)),  
 ('clf', LogisticRegression(solver='saga', max\_iter=1000))  
])  
pipeline.fit(X\_train, y\_train)

## 8. Validation Evaluation

y\_pred = pipeline.predict(X\_val)  
print(classification\_report(y\_val, y\_pred))  
cm = confusion\_matrix(y\_val, y\_pred)  
plt.matshow(cm)  
plt.show()

## 9. Error Analysis

val\_df = pd.DataFrame({'text': X\_val, 'true': y\_val, 'pred': y\_pred})  
mis = val\_df[val\_df['true'] != val\_df['pred']].sample(10, random\_state=42)  
print(mis)

## 10. Hyperparameter Tuning

param\_grid = {  
 'tfidf\_\_ngram\_range': [(1,1),(1,2)],  
 'clf\_\_C': [0.1,1,10]  
}  
grid = GridSearchCV(pipeline, param\_grid, cv=3, scoring='f1\_macro', n\_jobs=-1)  
grid.fit(X\_train, y\_train)  
print('Best params:', grid.best\_params\_)

## 11. Final Model Evaluation

best = grid.best\_estimator\_  
y\_pred\_best = best.predict(X\_val)  
print(classification\_report(y\_val, y\_pred\_best))

## 12. Test Set Predictions & Submission

test = pd.read\_csv('test\_data.csv')  
test['clean\_text'] = test['text'].apply(preprocess\_text)  
probs = best.predict\_proba(test['clean\_text'])  
sub = pd.DataFrame(probs, columns=best.classes\_)  
sub.insert(0, 'id', test['id'])  
sub.to\_csv('sample\_submission.csv', index=False)  
sub.head()