Amazon Mobile Reviews for Top 20 Brands

# Table of Contents

1. Installation  
2. Data Description  
3. Data Analysis  
4. Data Cleaning  
5. Model Training and Evaluation  
6. Results  
7. Contributing  
8. License

# Installation

To use the code in this repository, ensure you have the following libraries installed:  
  
pandas  
numpy  
matplotlib  
seaborn  
scikit-learn  
beautifulsoup4  
nltk  
  
You can install these libraries using pip:  
  
pip install pandas numpy matplotlib seaborn scikit-learn beautifulsoup4 nltk

# Data Description

The dataset used in this project is Amazon\_Unlocked\_Mobile.csv. It contains reviews of unlocked mobile phones sold on Amazon. The main columns of interest are:  
- Brand Name: The brand of the mobile phone.  
- Product Name: The name of the mobile phone.  
- Reviews: The text of the review.  
- Rating: The rating given by the reviewer.

# Data Analysis

1. Summary Statistics:  
print("Summary statistics of numerical features :   
", df.describe())

2. Distribution of Ratings:  
plt.figure(figsize=(12,8))  
df["Rating"].value\_counts().sort\_index().plot(kind="bar")  
plt.title("Distribution of Rating")  
plt.xlabel("Rating")  
plt.ylabel("Count")

3. Number of Reviews for Top 20 Brands:  
plt.figure(figsize=(12,8))  
brands[:20].plot(kind="bar")  
plt.title("Number of Reviews for Top 20 Brands")

4. Distribution of Review Length:  
plt.figure(figsize=(12,8))  
review\_length.loc[review\_length < 1500].hist()  
plt.title("Distribution of Review Length")  
plt.xlabel("Review length (Number of characters)")  
plt.ylabel("Count")

# Data Cleaning

The text data is cleaned by removing HTML tags, non-character symbols, stopwords, and by performing stemming. The cleaned text is then used for further analysis.  
  
def cleanText(raw\_text, remove\_stopwords=False, stemming=False, split\_text=False):  
 text = BeautifulSoup(raw\_text, "lxml").get\_text() # remove HTML  
 letters\_only = re.sub("[^a-zA-Z]", " ", text) # remove non-character  
 words = letters\_only.lower().split() # convert to lower case  
   
 if remove\_stopwords: # remove stopwords  
 stops = set(stopwords.words("english"))  
 words = [w for w in words if not w in stops]  
   
 if stemming: # perform stemming  
 stemmer = SnowballStemmer("english")  
 words = [stemmer.stem(w) for w in words]  
   
 return " ".join(words)

# Model Training and Evaluation

The cleaned data is split into training and testing sets. A TfidfVectorizer is used to convert the text data into numerical features. A Logistic Regression model is then trained and evaluated.  
  
# Logistic Regression  
lr = LogisticRegression()  
lr.fit(X\_train\_tfidf, y\_train)  
  
# Evaluate the model  
predictions = lr.predict(tfidf.transform(X\_test\_cleaned))  
modelEvaluation(predictions)

# Results

The model's performance is evaluated using accuracy, ROC-AUC score, classification report, and confusion matrix.  
  
def modelEvaluation(predictions):  
 print("\nAccuracy on validation set: {:.4f}".format(accuracy\_score(y\_test, predictions)))  
 print("\nROC-AUC score: {:.4f}".format(roc\_auc\_score(y\_test, predictions)))  
 print("\nClassification report:\n", metrics.classification\_report(y\_test, predictions))  
 print("\nConfusion Matrix:\n", metrics.confusion\_matrix(y\_test, predictions))

# Contributing

Contributions are welcome! Please create an issue or submit a pull request for any improvements.

# License

This project is licensed under the MIT License. See the LICENSE file for details.