**Overview**

This code performs sentiment analysis on Amazon product reviews. It involves handling class imbalance, implementing and evaluating various classifiers, and exploring topic modeling techniques. The entire process includes data loading, preprocessing, class balancing, model training, evaluation, hyperparameter tuning, and topic modeling.

**1. Importing Necessary Libraries**

* **pandas**: Used for data manipulation and analysis. It helps in loading and processing the dataset.
* **matplotlib.pyplot and seaborn**: Used for data visualization. Matplotlib provides plotting capabilities, while Seaborn is built on Matplotlib and provides a high-level interface for drawing attractive statistical graphics.
* **imblearn.over\_sampling.SMOTE**: Synthetic Minority Over-sampling Technique (SMOTE) is used to handle class imbalance by generating synthetic samples for the minority class.
* **sklearn.feature\_extraction.text.TfidfVectorizer**: Converts text data into numerical format using Term Frequency-Inverse Document Frequency (TF-IDF) representation.
* **sklearn.naive\_bayes.MultinomialNB**: Implements the Multinomial Naive Bayes algorithm for classification.
* **sklearn.model\_selection.train\_test\_split**: Splits the dataset into training and testing sets.
* **sklearn.metrics**: Provides functions for evaluating model performance (e.g., classification report, accuracy score, ROC AUC score).
* **sklearn.svm.SVC**: Implements Support Vector Classification.
* **sklearn.preprocessing.LabelEncoder**: Encodes target labels with value between 0 and n\_classes-1.
* **tensorflow.keras**: Used for building and training deep learning models.
* **xgboost.XGBClassifier**: Implements the XGBoost algorithm for classification.
* **sklearn.ensemble.VotingClassifier:** Combines multiple classifiers into a single ensemble model.
* **sklearn.decomposition.LatentDirichletAllocation and NMF:** Implements topic modeling techniques.

**2. Load the Dataset**

file\_path = "C:\\Users\\PRANAY\\Downloads\\train\_data.csv"

data = pd.read\_csv(file\_path)

The dataset is loaded from a CSV file into a pandas DataFrame.

**3. Data Exploration and Visualization**

* **Display the first few rows and basic information about the dataset:**

print(data.head())

print(data.info())

Provides an initial look at the data and its structure.

* **Check and visualize the distribution of sentiment categories:**

print(data['sentiment'].value\_counts())

plt.figure(figsize=(10, 6))

sns.countplot(x='sentiment', data=data)

plt.title('Distribution of Sentiment Categories')

plt.xlabel('Sentiment')

plt.ylabel('Count')

plt.show()

Helps understand the class distribution and identify any imbalance.

**4. Text Preprocessing**

* **Separate features and labels:**

X = data['reviews.text']

y = data['sentiment']

Splits the dataset into input features (reviews) and target labels (sentiment).

* **Convert text data to numeric using TF-IDF vectorizer:**

tfidf\_vectorizer = TfidfVectorizer(max\_features=5000)

X\_tfidf = tfidf\_vectorizer.fit\_transform(X)

Transforms the text data into TF-IDF features, limiting to 5000 most important features.

**5. Handling Class Imbalance**

* **Apply SMOTE to balance the classes:**

smote = SMOTE()

X\_balanced, y\_balanced = smote.fit\_resample(X\_tfidf, y)

print(pd.Series(y\_balanced).value\_counts())

Balances the dataset by generating synthetic samples for the minority class using SMOTE.

**6. Train-Test Split**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_balanced, y\_balanced, test\_size=0.2, random\_state=42)

Splits the balanced dataset into training and testing sets.

**7. Model Training and Evaluation**

* **Multinomial Naive Bayes:**

nb\_model = MultinomialNB()

nb\_model.fit(X\_train, y\_train)

y\_pred = nb\_model.predict(X\_test)

print("Multinomial Naive Bayes Accuracy:", accuracy\_score(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

y\_proba = nb\_model.predict\_proba(X\_test)

print("AUC-ROC Score:", roc\_auc\_score(y\_test, y\_proba, multi\_class='ovr'))

Trains and evaluates the Multinomial Naive Bayes model, printing accuracy, classification report, and ROC AUC score.

* **Support Vector Machine (SVM):**

svm\_model = SVC(kernel='linear', C=1.0, random\_state=42)

svm\_model.fit(X\_train, y\_train)

y\_pred\_svm = svm\_model.predict(X\_test)

print("Multi-class SVM Accuracy:", accuracy\_score(y\_test, y\_pred\_svm))

print(classification\_report(y\_test, y\_pred\_svm))

Trains and evaluates the SVM model, printing accuracy and classification report.

**8. Neural Network Models**

* **Feedforward Neural Network:**

label\_encoder = LabelEncoder()

y\_train\_encoded = label\_encoder.fit\_transform(y\_train)

y\_test\_encoded = label\_encoder.transform(y\_test)

y\_train\_categorical = to\_categorical(y\_train\_encoded)

y\_test\_categorical = to\_categorical(y\_test\_encoded)

model = Sequential([

Dense(128, activation='relu', input\_shape=(X\_train.shape[1],)),

Dropout(0.5),

Dense(64, activation='relu'),

Dense(len(label\_encoder.classes\_), activation='softmax')

])

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

history = model.fit(X\_train.toarray(), y\_train\_categorical, epochs=20, batch\_size=32, validation\_split=0.1, callbacks=[EarlyStopping(patience=3)])

\_, accuracy = model.evaluate(X\_test.toarray(), y\_test\_categorical)

print("Feedforward Neural Network Accuracy:", accuracy)

Trains a feedforward neural network and evaluates its performance.

* **LSTM Neural Network:**

X\_train\_lstm = np.reshape(X\_train.toarray(), (X\_train.shape[0], 1, X\_train.shape[1]))

X\_test\_lstm = np.reshape(X\_test.toarray(), (X\_test.shape[0], 1, X\_test.shape[1]))

model\_lstm = Sequential([

LSTM(64, input\_shape=(1, X\_train.shape[1]), activation='relu', return\_sequences=True),

Dropout(0.5),

LSTM(32, activation='relu'),

Dense(32, activation='relu'),

Dense(len(label\_encoder.classes\_), activation='softmax')

])

model\_lstm.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

history\_lstm = model\_lstm.fit(X\_train\_lstm, y\_train\_categorical, epochs=20, batch\_size=32, validation\_split=0.1, callbacks=[EarlyStopping(patience=3)])

\_, accuracy\_lstm = model\_lstm.evaluate(X\_test\_lstm, y\_test\_categorical)

print("LSTM Model Accuracy:", accuracy\_lstm)

Trains a Long Short-Term Memory (LSTM) neural network and evaluates its performance.

**9. Ensemble Models**

* **Multinomial Naive Bayes and XGBoost:**

nb\_model = MultinomialNB()

xgb\_model = XGBClassifier(n\_estimators=100, random\_state=42)

nb\_model.fit(X\_train\_resampled, y\_train\_resampled)

xgb\_model.fit(X\_train\_resampled, y\_train\_resampled)

nb\_pred = nb\_model.predict(X\_test)

xgb\_pred = xgb\_model.predict(X\_test)

nb\_accuracy = accuracy\_score(y\_test\_encoded, nb\_pred)

xgb\_accuracy = accuracy\_score(y\_test\_encoded, xgb\_pred)

print("Multinomial Naive Bayes Accuracy:", nb\_accuracy)

print("XGBoost Accuracy:", xgb\_accuracy)

Trains and evaluates Multinomial Naive Bayes and XGBoost models separately.

* **Voting Classifier:**

voting\_clf = VotingClassifier(estimators=[('nb', nb\_model), ('xgb', xgb\_model)])

voting\_clf.fit(X\_train\_resampled, y\_train\_resampled)

ensemble\_accuracy = voting\_clf.score(X\_test, y\_test\_encoded)

print("Ensemble Model Accuracy:", ensemble\_accuracy)

Combines the Naive Bayes and XGBoost models into an ensemble using voting and evaluates its performance.

**10. Feature Engineering**

* **Sentiment Score Feature:**

data['sentiment\_score'] = data['sentiment'].apply(lambda x: 1 if x == 'positive' else 0 if x == 'negative' else 0.5)

sentiment\_score\_sparse = sp.csr\_matrix(data['sentiment\_score'].values.reshape(-1, 1).astype(float))

X\_combined = sp.hstack((X\_tfidf, sentiment\_score\_sparse))

X\_train\_combined, X\_test\_combined, y\_train\_combined, y\_test\_combined = train\_test\_split(X\_combined, data['sentiment'], test\_size=0.2, random\_state=42)

nb\_model\_combined = MultinomialNB()

nb\_model\_combined.fit(X\_train\_combined, y\_train\_combined)

y\_pred\_combined = nb\_model\_combined.predict(X\_test\_combined)

print("Combined Feature Accuracy:", accuracy\_score(y\_test\_combined, y\_pred\_combined))

print(classification\_report(y\_test\_combined, y\_pred\_combined))

Adds a new feature representing the sentiment score and evaluates its impact on the model performance.

**11. Hyperparameter Tuning**

* **Grid Search for Hyperparameter Tuning:**

from sklearn.model\_selection import GridSearchCV

param\_grid = {'alpha': [0.1, 0.5, 1.0]}

grid\_search = GridSearchCV(MultinomialNB(), param\_grid, cv=5, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

best\_model = grid\_search.best\_estimator\_

y\_pred\_grid = best\_model.predict(X\_test)

print("Grid Search Best Parameters:", grid\_search.best\_params\_)

print("Grid Search Best Accuracy:", accuracy\_score(y\_test, y\_pred\_grid))

Uses grid search to find the best hyperparameters for the Naive Bayes model.

**12. Topic Modeling**

* **Latent Dirichlet Allocation (LDA):**

lda = LatentDirichletAllocation(n\_components=5, random\_state=42)

lda.fit(X\_tfidf)

topic\_distributions = lda.transform(X\_tfidf)print("LDA Topic Distributions:", topic\_distributions)

Applies LDA for topic modeling and prints the topic distributions.

* **Non-negative Matrix Factorization (NMF):**

nmf = NMF(n\_components=5, random\_state=42)

nmf.fit(X\_tfidf)

topic\_distributions\_nmf = nmf.transform(X\_tfidf)

print("NMF Topic Distributions:", topic\_distributions\_nmf)

Applies NMF for topic modeling and prints the topic distributions.

This comprehensive code covers data preprocessing, class balancing, model training, evaluation, feature engineering, hyperparameter tuning, and topic modeling for sentiment analysis on Amazon product reviews.

**Summary**

This code provides a comprehensive approach to sentiment analysis, including data preprocessing, handling class imbalance, training and evaluating various classifiers, implementing ensemble models, and performing topic modeling. The code demonstrates the use of different machine learning algorithms and deep learning models, evaluates their performance, and explores feature engineering and topic modeling techniques to gain deeper insights from the data.