1. Introduction

This project aimed to build a classifier that predicts review scores of films based on textual content. The primary challenge was to extract meaningful features from textual data and numeric features while optimizing the model's accuracy. I focused on exploring patterns in the data, feature engineering, and parameter tuning to achieve the best possible performance.

The final model is an XGBoost classifier that uses a combination of TF-IDF vectors, numerical features, and advanced sentiment-based features extracted from the text.

2. Data Processing and Feature Engineering

2.1 Initial Observations

During the exploratory data analysis, I identified several patterns in the review data that helped guide my feature engineering strategy:

- **Textual Lengths**: Longer reviews seemed to indicate stronger opinions, both positive and negative. Thus, text length in terms of words and characters was noted as a valuable feature.
- **Sentiment Indicators**: Sentiment in the text correlated well with the review score, suggesting that reviews with positive language likely received higher scores.
- **Keyword Presence**: Certain keywords (e.g., "good," "bad," "recommend") were frequently used in reviews and could be indicative of the score.**(However, because after I adding Keyword Presence, the accuracy get even worse, then I delete Keyword Presence in my model feature.)**

2.2 Feature Engineering

Based on these observations, I engineered a combination of basic and advanced features to provide the model with richer context.

Basic Features

1. **Text Length**: Count of words and characters in both the review text and summary.
2. **Helpfulness Ratio:** Calculated as the ratio of HelpfulnessNumerator to HelpfulnessDenominator, indicating the perceived helpfulness of the review.

Advanced Features

1. **Sentiment Score and Intensity**: Using TextBlob, I extracted the sentiment polarity (a value between -1 and 1, representing negative to positive sentiment) of each review. Additionally, I calculated the absolute value of this score as a measure of sentiment intensity.
2. **Keyword Presence**: Keywords associated with high or low scores were identified and added as binary features, indicating the presence or absence of terms such as "good," "bad," "recommend," etc.
3. **Punctuation Counts**: The number of exclamation points and question marks were counted as indicators of emotional intensity or uncertainty.

By combining these features with TF-IDF vectorized representations of the text and summary, the model could leverage both structured metadata and unstructured textual content for improved predictions.

3. Model Selection and Tuning

3.1 Algorithm Choice: XGBoost Classifier

I chose XGBoost for its high performance and flexibility with large datasets. Moreover, it can handle both structured and unstructured data. XGBoost also provides extensive options for parameter tuning, which helps optimize the model for this task.

3.2 Parameter Tuning

Several rounds of parameter tuning were performed to balance model complexity with accuracy, while also avoiding overfitting.

I use three laptops to keep trying different parameters in order to get higher accuracy.

Parameters like: Number of Estimators/Learning Rate/Max Depth and Min Child Weight/Gamma.

The most useful Parameter is Number of Estimators and learning Rate. However, if the Number of Estimators is too high, the running time for the model is too long even if accuracy is better. I once set the Number of Estimators to 1000 and ran the model for 4 hours which is inefficient. Therefore it is also important to have a good balance between accuracy and running time since this project has a time pressure.

3.3 Model Training and Validation

The data was split into training and validation sets to evaluate model performance accurately. K-fold cross-validation was used in the tuning phase to confirm that the chosen parameters generalized well across multiple splits.

4. Special Tricks and Observations

4.1 Special Tricks

Special Tricks I used in this Project is I used Three laptops to run different parameters at the same time, so I can select better parameter settings after a lot of attempts. Moreover, adding **Sentiment Score and Intensity** into my feature did increase accuracy a lot.

4.2 Feature Selection

One of the most effective techniques was combining TF-IDF with sentiment-based features. TF-IDF provided essential information about word importance, while sentiment intensity and keyword presence gave additional context about the emotional tone. This multi-faceted feature representation helped the model differentiate between nuanced opinions in reviews.

4.3 Balancing Feature Engineering and Model Complexity

Increasing Max_depth and n_estimator initially improved accuracy but led to overfitting. By adjusting gamma and using sentiment-based features, I achieved a balance between model complexity and performance. These adjustments helped generalize the model while retaining the ability to capture complex patterns in the data.

4.4 Observing Keyword Impact

Including binary indicators for specific keywords improved the model’s accuracy significantly. The model performed better on reviews that contained common opinion words, showing that simple keywords are often enough to signal sentiment. This discovery informed my decision to add keyword presence as a feature.

5. Assumptions and Limitations

5.1 Assumptions

1. **Textual Sentiment Consistency**: I assumed that the sentiment polarity and intensity are consistent indicators of the review score.
2. **Keyword Relevance**: Another assumption was that the selected keywords hold consistent sentiment value across all reviews.

5.2 Limitations

1. **Keyword Sensitivity**: Not all keywords might be equally relevant across contexts. Some terms might carry different connotations depending on the product or service, which this model doesn't account for.
2. **Limited Generalizability to Other Domains**: The model was trained specifically on this dataset, and results might vary if applied to other datasets with different vocabulary, sentiment trends, or scoring criteria.

6. Conclusion and improvements in final projects

In conclusion, combining sentiment analysis, keyword detection, and text length features with traditional TF-IDF representations is useful. It allows the XGBoost model to gain a more comprehensive understanding of the review data. The results demonstrate that feature engineering plays a crucial role in text classification tasks, especially when sentiment and intensity are key indicators of the target variable.

Although the Xgboost model is a good model, it is too complicated and hard to use, especially when I need to make a file. It always goes wrong. Which is very annoying to me and time consuming. I will try different models such as random forests in my final project.

Moreover, it is important to know which keys are the most important feature to get the result score. Find those keys and use them in my model.

I got 0.53 accuracy on my first try. In my first model feature, I didn't include ‘LexicalRichness’ and ‘ExclamationCount’ , but those two features are really important in this project.

In my final project, my model will use the ‘macd indicator’ to predict stock price. I will use my economic knowledge to find important features and remove some disturbing features.

Oh! More importantly, I will never use a colab webpage to run models with massive samples. It is so annoying since the webpage will crash and every piece of data I made will be gone which did happen in this midterm project.

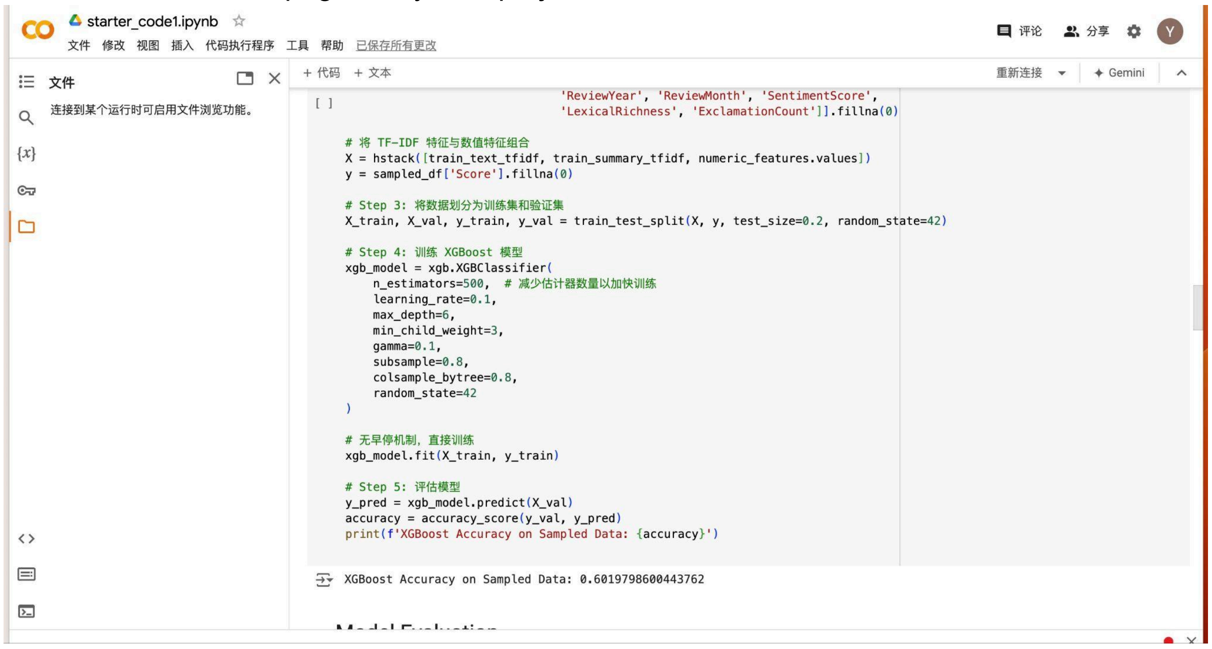
7. Further information(discussed with Professor Galletti)

Since I learned about colab in lab sessions and use colab webpage during labs and assignment 5, I started this project with colab webpage as well. In the beginning , everything was fine, and I successfully got a good result with a 0.6019798600443762 accuracy. I made a submission file and submitted it to a kaggle competition with good scores. However, the webpage crashes and deletes all the csv files and data. I have to use anaconda instead. However, even I copied and pasted the same code I wrote on colab, the accuracy just can't go back to 0.6019798600443762. I don't know why and have no idea why I have this problem.

The following picture is a webpage screenshot for colab.

I have discussed this problem with professor Galletti, he asked me to write in this file as will to report this problem.

One lesson I learned here is never use a webpage to handle large datasets. :(I will definitely not use the colab webpage in my final project.


starter_code1.ipynb
文件 修改 视图 插入 代码执行程序 工具 帮助 已保存所有更改
评论
分享
文件
连接到某个运行时可启用文件浏览功能。
+ 代码 + 文本
重新连接
Gemini

```
[ ]
                              'ReviewYear', 'ReviewMonth', 'SentimentScore',
                              'LexicalRichness', 'ExclamationCount']].fillna(0)

    # 将 TF-IDF 特征与数值特征组合
    X = hstack([train_text_tfidf, train_summary_tfidf, numeric_features.values])
    y = sampled_df['Score'].fillna(0)

    # Step 3: 将数据划分为训练集和验证集
    X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

    # Step 4: 训练 XGBoost 模型
    xgb_model = xgb.XGBClassifier(
        n_estimators=500,  # 减少估计器数量以加快训练
        learning_rate=0.1,
        max_depth=6,
        min_child_weight=3,
        gamma=0.1,
        subsample=0.8,
        colsample_bytree=0.8,
        random_state=42
    )

    # 无早停机制，直接训练
    xgb_model.fit(X_train, y_train)

    # Step 5: 评估模型
    y_pred = xgb_model.predict(X_val)
    accuracy = accuracy_score(y_val, y_pred)
    print(f'XGBoost Accuracy on Sampled Data: {accuracy}')
```

```
XGBoost Accuracy on Sampled Data: 0.6019798600443762
```