CA337 A5 Written Report   Cathal Ryan

Introduction

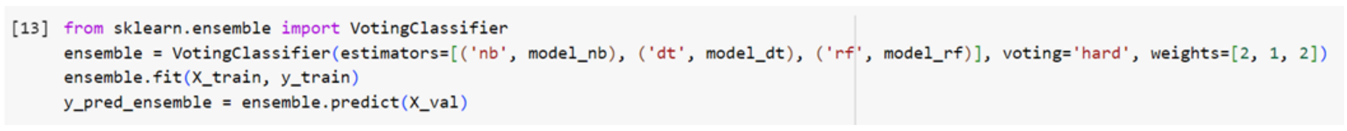
For this assignment I decided to do two ideas involving machine learning and one idea involving web app development. Regarding machine learning, the two ideas I decided to try were improving the ensemble accuracy and experimenting with sentiment lexicons. Also regarding web app development I decided to try and make a similar app to the one made in flask but in gradio. Therefore all my workings for this assignment were in the one jupyter notebook and it is all laid out in order, having my two machine learning ideas first and then my app in gradio.

Idea 1 – Improving the ensemble accuracy

In the task of enhancing the ensemble accuracy for sentiment analysis, I addressed the imdb dataset consisting of text samples labelled with sentiment scores (0 or 1) and aimed to boost the overall performance of an ensemble model. The dataset included diverse sentiments, varying from highly positive to strongly negative.

The initial ensemble was constructed using three distinct classifiers: Naive Bayes, Decision Tree, and Random Forest. Each classifier contributed its predictions, and the ensemble combined these individual outputs to form a final decision. To assess the ensemble's performance, I employed metrics such as accuracy and confusion matrices for individual classifiers as well as the ensemble.

The primary focus was on refining the ensemble accuracy through experimentation with hyperparameters and feature engineering. I systematically adjusted hyperparameters for each classifier, fine-tuning them to capture nuanced patterns in the data. I focused on adding weights to the three different models and giving extra weights to the two models with the higher accuracy score (Naive bayes and random forest). I found that this improved the accuracy score after previously giving extra weight to just naive bayes and then to just random forest but eventually found that giving both extra weight improved the ensembles overall accuracy.



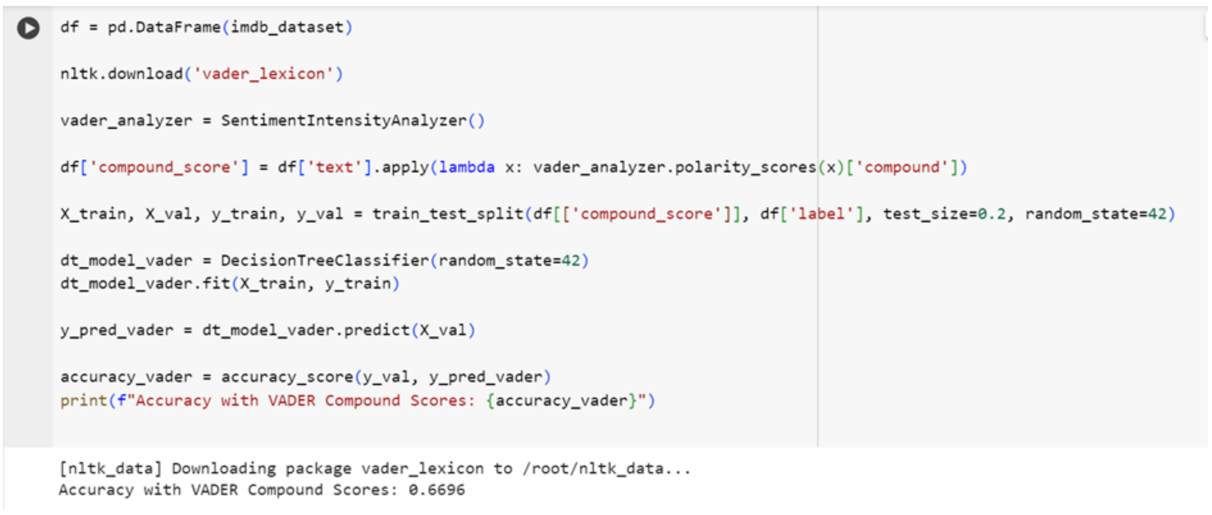
```
[13] from sklearn.ensemble import VotingClassifier
     ensemble = VotingClassifier(estimators=[('nb', model_nb), ('dt', model_dt), ('rf', model_rf)], voting='hard', weights=[2, 1, 2])
     ensemble.fit(X_train, y_train)
     y_pred_ensemble = ensemble.predict(X_val)
```

Idea 2 – Exploring the use of sentiment lexicons to improve classification accuracy

Additionally, I explored the impact of incorporating sentiment lexicons, such as VADER, to augment the models' understanding of sentiment nuances.The iterative process involved training and evaluating individual classifiers, analysing their weaknesses, and strategically adjusting parameters to enhance their contributions to the ensemble. Simultaneously, sentiment lexicons were incorporated to introduce an additional layer of semantic understanding, potentially capturing subtleties that conventional models might overlook.

The final ensemble, after rigorous optimization, while its accuracy didn't improve using VADER, I found that using TF-IDF improved the accuracy of the decision tree classifier. I decided to choose the decision tree classifier because it was the model with the lowest accuracy score and thought that it would make the most sense to try and improve its accuracy over the others.

When writing the code for the VADER lexicon, I first converted the data into a data frame using pandas. I then downloaded the VADER lexicon and initialised the VADER sentiment analysis. I then applied the VADER sentiment analysis to the text data. I split the data into training and validation sets. I then trained a decision tree classifier onto the compound scores. And finally I made predictions on the validation set and I evaluated the performance.



```
df = pd.DataFrame(imdb_dataset)

nltk.download('vader_lexicon')

vader_analyzer = SentimentIntensityAnalyzer()

df['compound_score'] = df['text'].apply(lambda x: vader_analyzer.polarity_scores(x)['compound'])

X_train, X_val, y_train, y_val = train_test_split(df[['compound_score']], df['label'], test_size=0.2, random_state=42)

dt_model_vader = DecisionTreeClassifier(random_state=42)
dt_model_vader.fit(X_train, y_train)

y_pred_vader = dt_model_vader.predict(X_val)

accuracy_vader = accuracy_score(y_val, y_pred_vader)
print(f"Accuracy with VADER Compound Scores: {accuracy_vader}")
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
Accuracy with VADER Compound Scores: 0.6696
```

Idea 3 – Getting an app running from one of my CA337 Jupyter notebooks using Gradio

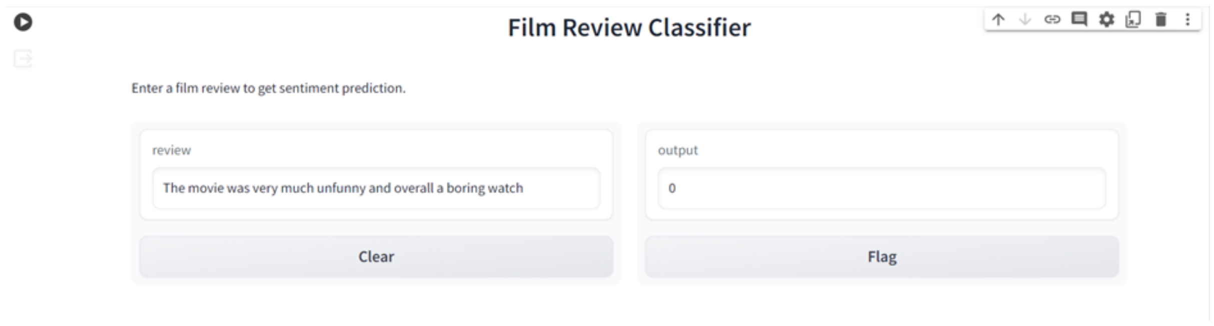
I decided to make a very similar app to the one made in flask but this time using gradio in jupyter notebooks. In creating the film review classifier using Gradio, the goal was to develop a user-friendly application that predicts the sentiment of a film review. This interactive tool allows users to input a review, and the system generates a prediction indicating whether the sentiment is positive or negative.

To begin, I used the imdb dataset once again. The dataset was split into training and testing sets to train a machine learning model. For simplicity, a basic model using a Naive Bayes classifier with TF-IDF vectorization was employed. This model was trained to recognize patterns in the text and make predictions based on the learned patterns.

Once the model was trained, Gradio was used to build an intuitive and accessible interface. The Gradio interface consists of a text input box where users can type or paste a film review. The model then processes the input using the trained classifier and promptly provides a prediction indicating whether the review expresses a positive or negative sentiment. This streamlined interface allows users to obtain sentiment predictions effortlessly.

The code for this application involves setting up the machine learning model, integrating it with Gradio, and launching the interface. The model is trained on a dataset to learn the patterns associated with positive and negative sentiments. Gradio then takes care of the user interface, providing a straightforward input box and a button for prediction.

The application's purpose is to serve as a practical tool for anyone curious about the sentiment of a film review. It doesn't require any coding knowledge, and users can interact with the model using plain language. This kind of application could be beneficial for movie enthusiasts, critics, or anyone interested in gauging the sentiments expressed in film reviews.


Film Review Classifier

Enter a film review to get sentiment prediction.

review: The movie was very much unfunny and overall a boring watch

output: 0

Personal Retrospective

Through this assignment, I have been able to understand and learn new techniques and also hone in and practice on techniques I have learned in this course already. I've gained insights into the integration of machine learning with Gradio, simplifying the deployment of sentiment analysis models. I once again had to train a basic sentiment classifier using a Naive Bayes algorithm and TF-IDF vectorization and also create and improve an ensemble and its accuracy. Gradio facilitated the creation of an interactive interface, which would allow users to input film reviews and receive sentiment predictions effectively. This assignment taught me the accessibility Gradio brings to machine learning applications, making complex models user-friendly. It highlights the potential for deploying machine learning solutions for broader audiences, emphasising the importance of user-friendly interfaces in bridging the gap between advanced algorithms and everyday users.