Project Name: NLP Harry Potter continuation of NLP Project
Project Type: optional solo project

Project Goals:
Continuing from partner NLP project. Using a different data set will results be better?
Does reducing the imbalance in data sampling improve performance?
Original Goals:
Can the programming language of a GitHub repository be predicted from the text of the README document?
Create a dataset of GitHub readme documents, Explore using NLP techniques, Build a model that predicts the programming language using NLP techniques to extract features and classification algorithms for prediction.
Deliverables: Well documented Jupyter Notebook

Stage	**Tools**	**Brief Description of Process**	**Challenge Resolution**
Plan	● Visual Studio	● Continuation of NLP Project ● What are the results with a different dataset? ● Does fixing data imbalance improve performance?	●
Acquire	● Visual Studio ● .py scripts	● .Using web-scrapping functions already built get a dataset for repos that include “Harry Potter” ●	● No unusual challenges in this section had already built functions
Prepare	● Visual Studio ● .py scripts	● Additional stop words added that were found to be common to all languages ● after 1st run added stop words of “harry”, “potter”, “run”,” file”, and “house” ● after 2nd run added stop words “&#9”, “use” ●	● No unusual challenges in this section
Explore	● Jupyter	● Javascript was over represented in the	● Used same code from

	Notebook ● Seaborn ● Matplotlib ● Stats ● Sklearn	dataset ● Visualized the top 20 most common words by language ● Visualed bi-grams by language	NLP project
Model	● Jupyter Notebook ● Sklearn ● Multiple ML models tested	● Used Bag of Words and TF-IDF as vetorizors ● Repeated model process and evaluation as previously ● Established baseline of 39% by predicting all would be most common = Javascript ● Logistic Regression and Random Forest performed about the same. ● Logistic Regression had 53% accuracy and Random Forrest had 51% accuracy	● Used same code from NLP project
Next Step	● Jupyter Notebook ● Sklearn	● Resampled data using SMOTE ● Accuracy was only slightly better on unseen data after balancing dataset ● HOWEVER, there was a 14% improvement in average F1 score after balancing	● Used same code from NLP project
Model Explanation	**How does your algorithm work?**	● Technically a regression algorithm (Goal is to find the values for the coefficients that weight each input variable). ● Used to predict binary outcomes. ● The output is a value between 0 and 1 that represents the probability of one class over the other	● No unusual challenges in this section