Skynet Project Post-Mortem

The purpose of this Project Post-Mortem (PPM) is to review the Skynet team's ("the team") project successes, challenges, and future developments while working on a Multi-Class Classification Model from April 7-13, 2023. This document will detail project methodology, problems, causes, and resolutions.

Methodology

Team Skynet selected a dataset containing ~220,000 rows of Tweet-data regarding ChatGPT. Following data pre-processing, cleansing, and Exploratory Data Analysis (EDA), the team selected a Natural Language Processing (NLP) Sentiment Analysis using a multi-class classification model to categorize Tweets into "bad," "neutral," or "good" sentiment classes.

The goal of the analysis was to correctly capture Tweet sentiment by achieving an accuracy above 80%. Achieving this goal would foster an understanding of how Twitter-users feel about ChatGPT. Team Skynet believes that, as technologists, we should be at the forefront of understanding use-cases for Large Language Models (LLM) like ChatGPT, to prepare for clientele that may be looking to take advantage of the latest technology.

Successes

1. Performance Improvement
    a. The team celebrated an initial success with a baseline accuracy of 61% using a SimpleTransformers Model, prior to processing and cleansing the data. This was an important success because the model type performed better than a 50/50 equal distribution, indicating potential for further model development and accuracy growth. The manual baseline accuracy also performed far better than our H20 AutoML method, which produced an accuracy of 44%.
    b. By only increasing the sample size by 5x, the group celebrated another win – growing model accuracy from 61% to 81%. This iteration still included no cleansing of the data.
2. Higher Data Quality
    a. The team processed the data to improve data quality going into the model. Data processing included removing punctuation, lower-casing the input values, removing emoji serials, and removal of hyperlinks. By improving the data quality, we improved our accuracy from 81% to 84%.
3. Save Time
    a. Team Skynet increased the Jupyter Notebook and instance sizes to run the model quicker. With this great achievement, the team was able to run the entire dataset achieving the best model accuracy at 92%. Overall, we were able to improve model accuracy by 50%.

Challenges & Limitations

1. Tweet Data
    a. Though we were able to successfully predict sentiment at a 92% accuracy in this model, we need to be extremely aware of the limitation that this accuracy was only trained on Twitter-specific data. Information on ChatGPT that may be in the news, on another social media platform, or blog will not be included in this sentiment analysis.
2. Limited Time
    a. The data was only collected for a short period of time from November 2022. So, it is important to consider that current events at this time are especially relevant, as they may have had impact on tweet-goers thoughts/feelings surrounding ChatGPT. A longer time period of tweet data would be helpful but may slow the process of cleansing the data and running the model.
    b. It is essential that we also bring attention to the limited time (~5 days) Team Skynet had to select data, assess data pre-processing needs, choose a machine learning model, implement the model, and construct necessary documentation.
3. Sentiment Limitation
    a. The data that the team selected was pre-populated with sentiment by the collector. We are unsure how the sentiments were selected by the collector, or what methodologies were used. If time allowed, the team would have preferred to construct our own algorithm for defining sentiment categories.
4. Package Restrictions
    a. Unfortunately, the package that our team selected was not supported to deploy on AWS Sagemaker as an endpoint. Please review future developments for additional information.

Future Developments

1. Scale
    a. Eventually, we aim to generalize and scale the model to capture sentiment about new technologies, outside of ChatGPT.
2. Deploy
    a. In the event time would allow, we would Take the software stack (environment to run the model and the code) > Load it into a Docker > Run on and EC2 Instance > Load to a Webserver (add to the software stack to send the results through the API endpoint) > Run on the Webserver > Load the Model and take the API Request > Then the Webserver would use the model to calculate response and send a response back.