**Approach and Methodology**

During this project, I analyzed employee email data to understand sentiment trends and identify potential flight risks using Python. I utilized the transformers, pandas, seaborn, matplotlib, sklearn, and numpy Python libraries in my code. I mainly used pandas to handle the employee email data frame I created to complete the tasks below. First, I called a variation of a BERT NLP (Natural Language, Processing) model I found to simplify the classification process using the pipeline module in the transformers library. I inputted the data through the model and created a standardized new for with a sentiment column (“Positive”, “Negative”, “Neutral”).

Next, I cleaned and explored the dataset using descriptive statistics and visualizations with the seaborn library. Then, I created an iteration with pandas to first assign a numerical score (-1, 0, 0) to the sentiment column and calculated monthly sentiment scores per employee. I found the top positive and negative sentiment senders for every month and stored the output in a pivot table CSV file for later processing (top\_negative\_employees.csv and top\_positive\_employees.csv). Moving forward, I detected employees with four or more negative scores over a rolling 30-day period and flagged them as potential flight risks. I exported the list of employees considered flight risk into a CSV file (flight\_risk\_employees).

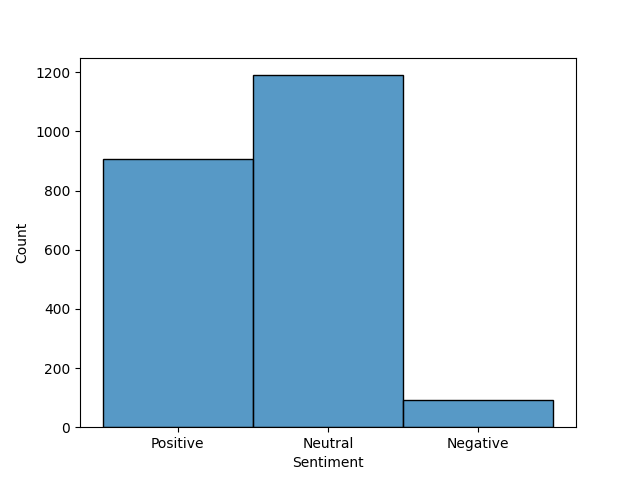
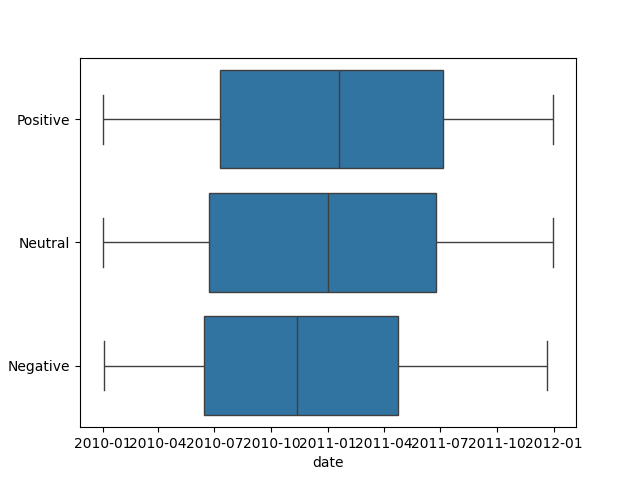
I used the sklearn library to create a linear regression model and trained the model to predict the sentiment scores, which were numerically encoded. I engineered several time-based and behavioral features such as message frequency, lagged sentiment, and message length. The model performance was evaluated using MSE and R². While most features showed weak predictive power, cumulative sentiment average and word count provided relatively stronger signals.

My approach combines NLP, exploratory analysis, behavioral scoring, and predictive modeling to surface insights into employee sentiment. While the linear model had limited accuracy, the methodology highlights useful patterns and lays the groundwork for more advanced modeling.

**Exploratory Data Analysis Key Findings**

The exploratory data analysis (EDA) revealed several key insights about the sentiment distribution within the dataset. Interestingly, the dataset did not have any duplicate or null values that needed to be cleaned. The dataframe had “Subject”, “body”, and “date” as object types, while “date” is datetime64[ns]. Sentiment was processed as a category type to make it easier to process when generating the plots.

After generating plots to visualize the relationships between the columns, I found that the distribution of sentiment over time (by the month year) showed that the majority of emails exhibited a neutral sentiment, followed by a positive sentiment, and lastly a negative sentiment (See Figure A). I discovered that the negative sentiment tended to occur earlier in the dataset, while the positive sentiment was more prevalent in the later part (See Figure B). Additionally, January 2010 emerged as the month with the highest volume of emails, suggesting it as a particularly active period (See Figure C and D). December 2010 stood out as the month with the most negative sentiment (See Figure C and D). October 2011 stood out as an outlier with equal amounts of negative and positive sentiments, indicating a period of mixed emotions among the senders (See Figure C and D).

Figure AFigure B

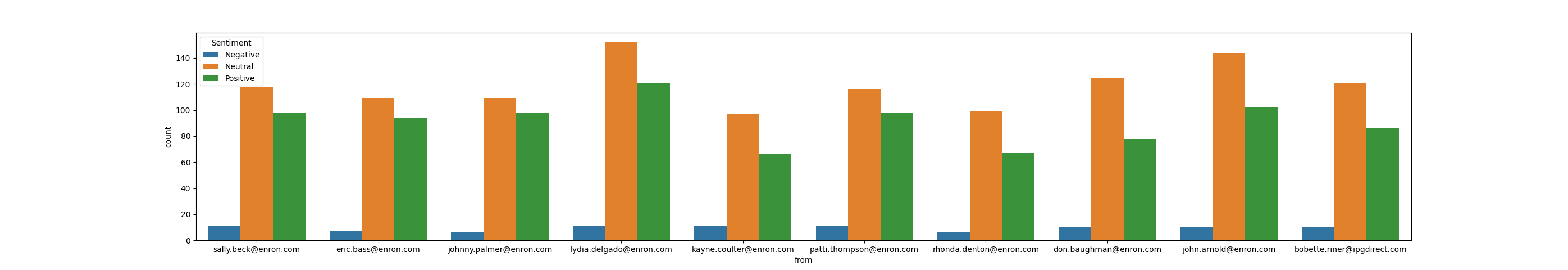


Figure C

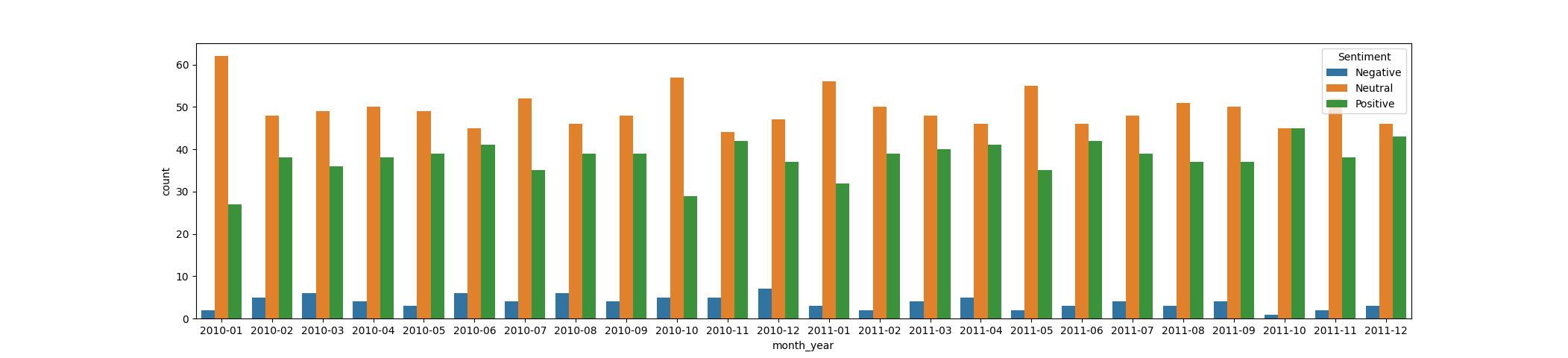
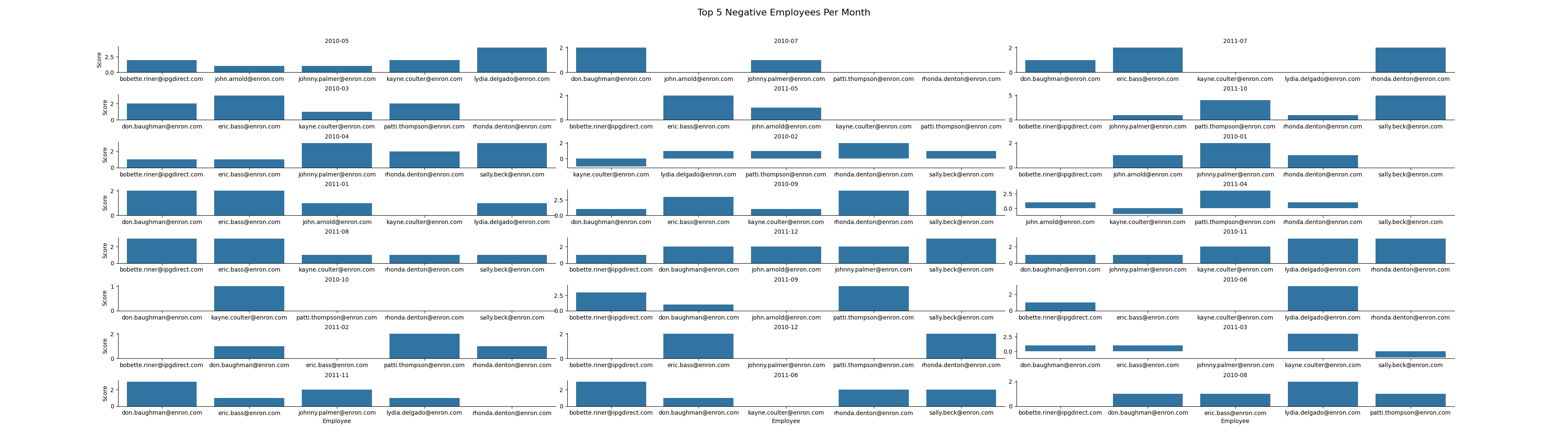
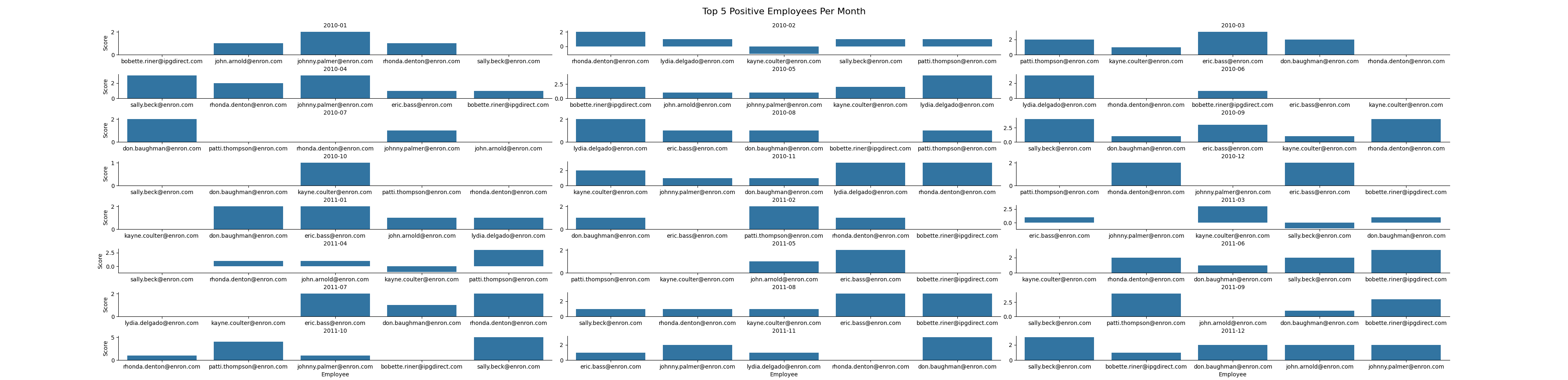


Figure D

**Employee Scoring and Ranking Process**

During the employee scoring and ranking process, I used the pandas library to achieve the iterations and access the dataframe. My first step was to group the dataset by month using the month\_year period. Each employee’s monthly sentiment score was calculated by assigning a value of +1 for positive messages and -1 for negative ones for each employee. Neutral emails are excluded from scoring. For every month, a monthly\_scores dictionary stores each employee’s cumulative sentiment score. The monthly\_scores dictionary was sorted to identify the top positive and negative employees with the results being stored in two lists respectively (top\_positive\_employees and top\_negative\_employees). These results are then organized into structured data tables that can be easily analyzed to track sentiment changes and potential employee engagement risks.

See the visualized output below.

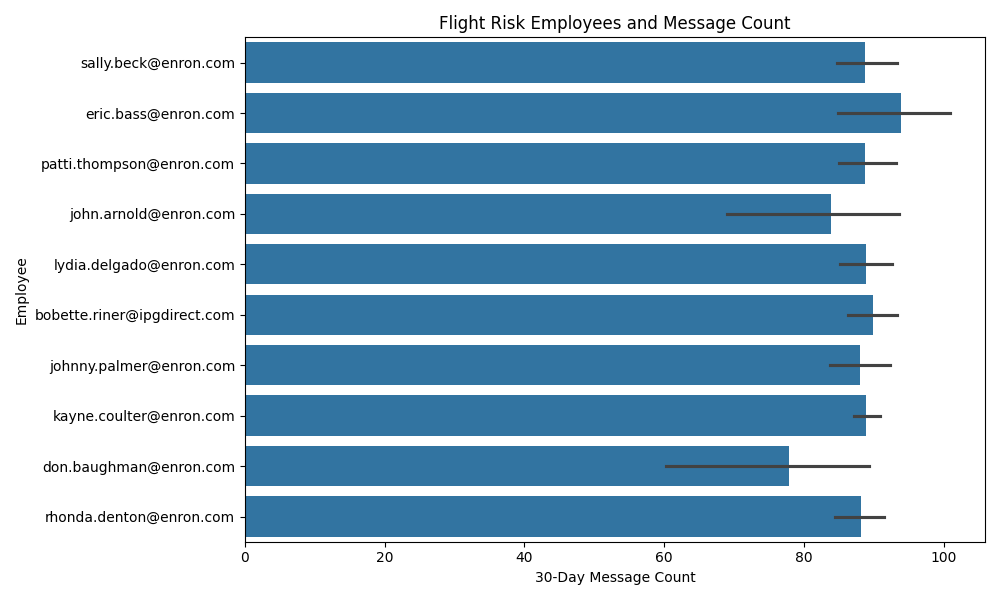


**Flight Risk Identification Criteria and Outcomes**

The process for the flight risk identification was very straightforward. The criteria for an employee to be put on the list were that they had to have 4 or more emails with negative sentiment over a 30-day rolling window. I used the .rolling() function and stored the 30-day time frame in a variable which counted the frequency of the emails during that window. Then, I created an iteration to find each employee whose count was equal to or greater than four and stored them in a list. I took measures to exclude all duplicates for this list.

The outcome of the script showed that the employees at risk are Don Baughman, John Arnold, Kayne Coulter, Sally Beck, Bobette Riner, Johnny Palmer, Lydia Delgado, Patti Thompson, Eric Bass, and Ronda Denton.

I exported a bar plot of the employee and their scores.



**Overview and Evaluation of the Predictive Model**

I used the sklearn library to develop a linear regression model. The standard of an ideal predictive model I used for this model is when the mean square error is MSE < 0.22 with a positive R-squared of > 0.2. The R-squared was lowered because the string body texts have a high variability due to the unstructured text, different contexts, and other noise.

I engineered several features to train with the model:

Message\_Frequency measured the amount of emails sent by the date for each employee.

Message\_Frequency\_MonthYear found the amount of emails sent by the month and year for each employee.

Day\_of\_week found the day of the week for the messages.

Prev\_Sentiment\_SameSender found the sentiment for the sender’s previous email.

Prev\_Sentiment\_2\_SameSender found the sentiment for the sender’s second to last email.

Prev\_Sentiment\_3\_SameSender found the sentiment for the sender’s third to last email.

Time\_Since\_Last\_Message was the time difference from the previous message.

Msg\_Count\_Rolling30 found the number of messages for each user per every 30 day window regardless of the month and year.

Message\_Length found the character length of each email’s body.

Word\_Count found the word length of each email’s body.

Cumulative\_Sentiment\_Avg found the cumulated average of the sentiment score.

When I ran these features through the model, the results were very poor despite the array of features I used.

Message\_frequency had little predictive value with a high MSE of and negative R-squared along with a positive coefficient with Sentiment. I created Message\_Frequency\_MonthYear to see if it had a better predictive value than Message\_Frequency for specific days. I ideally wanted the Month\_Year to return more frequencies per employee. However, the results were the same as Message\_Frequency.

Day\_of\_week had little predictive value with a high MSE and negative R-squared along with a positive coefficient with Sentiment. Msg\_Count\_Rolling30 had little predictive value with a high MSE and negative R-squared along with a positive coefficient Time\_Since\_Last\_Message had little predictive value with a high MSE and negative R-squared along with a negative coefficient with Sentiment.

Prev\_Sentiment\_SameSender had little predictive value with a high MSE and negative R-squared along with a negative coefficient with Sentiment. I created Prev\_Sentiment\_2\_SameSenderSame and Prev\_Sentiment\_3\_SameSender to see if the sentiments from two messages back and three messages back, respectively, would affect future sentiment. However, there was the same issue with high MSE and a negative R-squared when I ran 2 and 3 separately and ran all three together.

After a lot of trial and error, I was finally able to get a positive R-squared value which showed that some features did have an effect on sentiment scores.

Message\_Length had a much better predictive value with a MSE of .31 and positive R-squared of 0.065 along with a positive coefficient 0.0005 with Sentiment. Word\_Count had a much better predictive value with a MSE of .31 and positive R-squared of 0.075 along with a positive coefficient 0.003 with Sentiment Cumulative\_Sentiment\_Avg had a much better predictive value with a MSE of .31 and positive R-squared of 0.024 along with a positive coefficient 1.138899 with Sentiment.

Word\_Count had the highest predictability with the lowest MSE and highest R-Squared value. There was also a positive coefficient. Therefore, as the word count increases, the more likely the message is to be positive. Message\_Length had the second highest predictability. There was also a positive coefficient. Therefore, as the message length increases, the more likely the message is to be positive. The Cumulative\_Sentiment\_Avg had the third highest predictability. This feature also had the strongest positive coefficient. Therefore, the higher average sentiment is likely to predict more positive future sentiments.

Overall, the linear regression model needs to be trained more on better features and combinations of features with positive R-squared scores. The negative R-squared scores meant that most of the features had less than no effect on sentiment and could not predict the scores. The high MSE scores meant that there is more than the ideal amount of errors in predicting sentiment. The majority of the features showed poor MSE and R-squared numbers which indicate that either the model needs better features or that the model is not the best fit to predict sentiment. I suggest using a more complex model in order to achieve better predictability. In the future, more work on this model is needed.