UPDATED REPORT

Employee Sentiment Analysis

Abstract

This project involves a pipeline that processes employee email messages to determine their sentiment (positive, neutral, or negative) and uses that sentiment to derive multiple business insights

K Shreyank

kjshreyank@gmail.com

Updated

AI – Project – Submission - Employee Sentiment Analysis

Aim –

To analyze employee email communication data and extract meaningful insights through sentiment analysis, exploratory data visualization, and predictive modeling — with the goal of identifying patterns, ranking employees by communication tone, and detecting potential flight risks based on negative communication trends.

Summary –

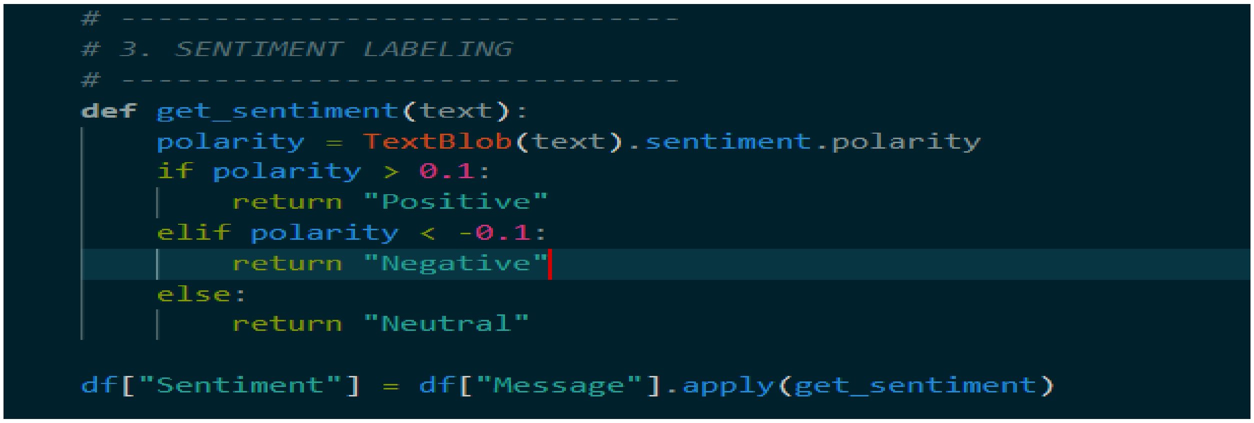
This project involves a pipeline that processes employee email messages to determine their sentiment (positive, neutral, or negative) and uses that sentiment to derive multiple business insights:

- Sentiment labeling is done using TextBlob.
- Visualizations are created to understand the distribution and trends of sentiments.
- Employees are ranked based on negative communication patterns.
- A rolling window mechanism is used to identify employees with consistently negative communication, flagging them as potential flight risks.
- A regression model is built to predict message length based on text features.

Explanation of Features –

1. **Sentiment Labeling**

- Uses TextBlob to analyze the sentiment polarity of each email message.
- Labels messages as: Positive (polarity > 0.1), Negative (polarity < -0.1) and Neutral (in between)



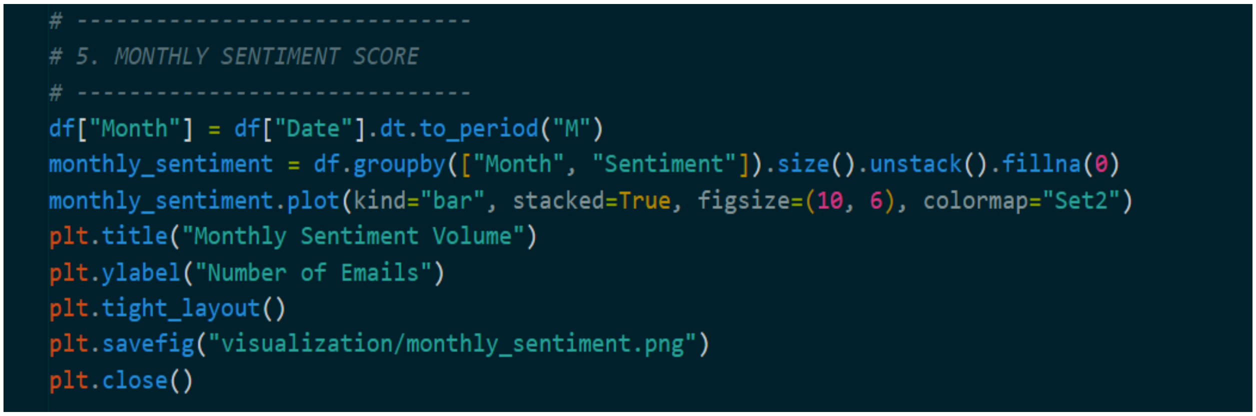
```
# -------------------------------
# 3. SENTIMENT LABELING
# -------------------------------
def get_sentiment(text):
    polarity = TextBlob(text).sentiment.polarity
    if polarity > 0.1:
        return "Positive"
    elif polarity < -0.1:
        return "Negative"
    else:
        return "Neutral"

df["Sentiment"] = df["Message"].apply(get_sentiment)
```

2. Exploratory Data Analysis (EDA) & Visualizations

- Bar chart of sentiment distribution shows the number of emails per sentiment type.
- Helps identify overall communication climate across employees.

3. Monthly Sentiment Scoring

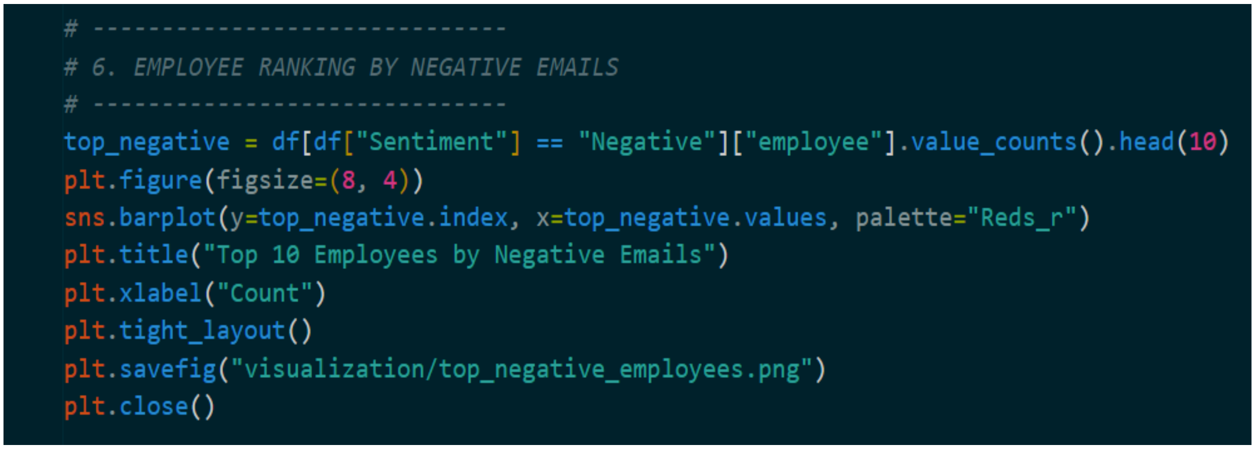
- Emails are grouped by month and sentiment.
- A stacked bar chart visualizes changes in sentiment over time.
- Useful for tracking communication mood and behavioral shifts.



```
# -----------------------------
# 5. MONTHLY SENTIMENT SCORE
# -----------------------------
df["Month"] = df["Date"].dt.to_period("M")
monthly_sentiment = df.groupby(["Month", "Sentiment"]).size().unstack().fillna(0)
monthly_sentiment.plot(kind="bar", stacked=True, figsize=(10, 6), colormap="Set2")
plt.title("Monthly Sentiment Volume")
plt.ylabel("Number of Emails")
plt.tight_layout()
plt.savefig("visualization/monthly_sentiment.png")
plt.close()
```

4. Employee Ranking

- Employees are ranked based on how many negative emails they've sent.
- Top 10 are displayed using a horizontal bar chart.
- Allows management to quickly identify frequently negative communicators.

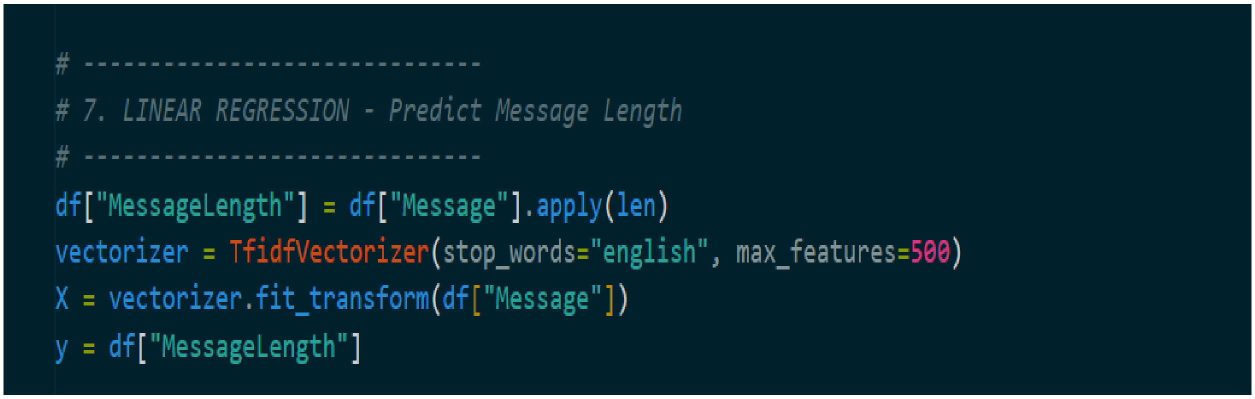


```
# -----------------------------
# 6. EMPLOYEE RANKING BY NEGATIVE EMAILS
# -----------------------------
top_negative = df[df["Sentiment"] == "Negative"]["employee"].value_counts().head(10)
plt.figure(figsize=(8, 4))
sns.barplot(y=top_negative.index, x=top_negative.values, palette="Reds_r")
plt.title("Top 10 Employees by Negative Emails")
plt.xlabel("Count")
plt.tight_layout()
plt.savefig("visualization/top_negative_employees.png")
plt.close()
```

Updated

5. Feature Engineering

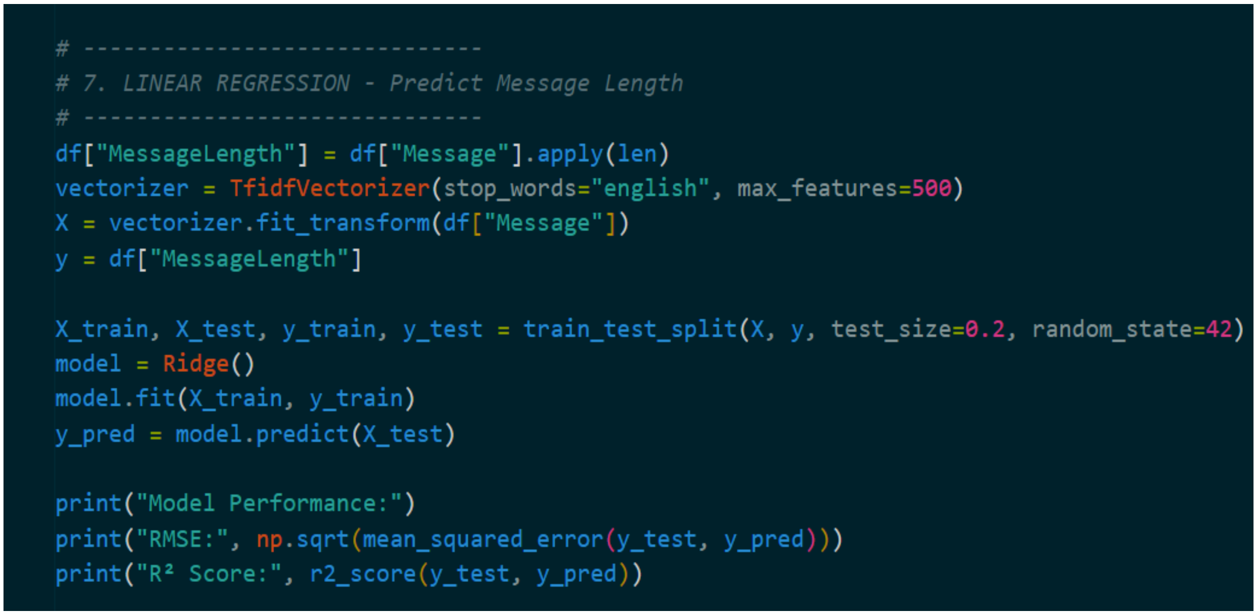
- Extracts numerical features from text:
- Message Length: character count
- Wordcount: number of words
- Supports both analysis and predictive modeling.



```
# ----------------------------
# 7. LINEAR REGRESSION - Predict Message Length
# ----------------------------
df["MessageLength"] = df["Message"].apply(len)
vectorizer = TfidfVectorizer(stop_words="english", max_features=500)
X = vectorizer.fit_transform(df["Message"])
y = df["MessageLength"]
```

6. Predictive Modeling

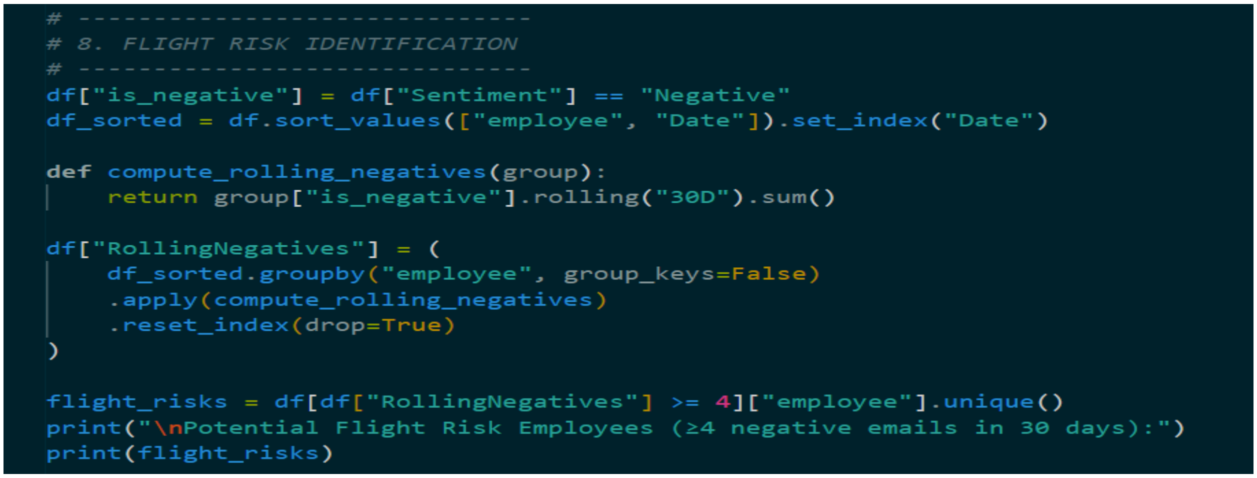
- A Ridge regression model is trained to predict email message length using TF-IDF vectorized features.
- Evaluates model with RMSE and R² Score.
- Demonstrates basic supervised learning using email text.



```
# ------------------------------
# 7. LINEAR REGRESSION - Predict Message Length
# ------------------------------
df["MessageLength"] = df["Message"].apply(len)
vectorizer = TfidfVectorizer(stop_words="english", max_features=500)
X = vectorizer.fit_transform(df["Message"])
y = df["MessageLength"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = Ridge()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Model Performance:")
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
print("R² Score:", r2_score(y_test, y_pred))
```

7. Flight Risk Identification

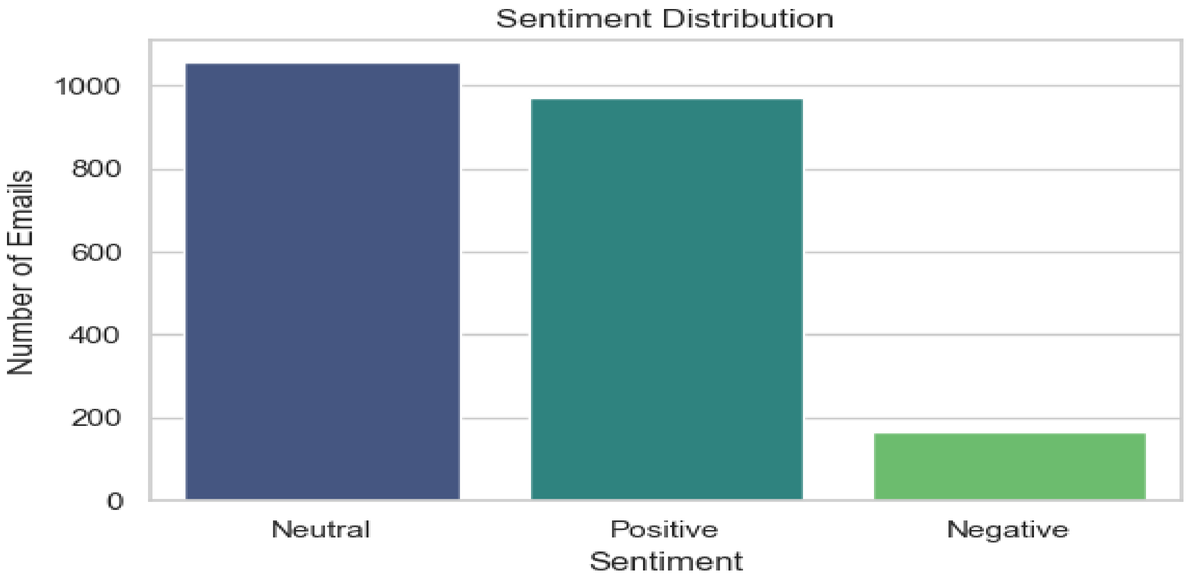
- For each employee, counts the number of negative emails in a rolling 30-day window.
- If this count is ≥ 4, the employee is flagged as flight risk.



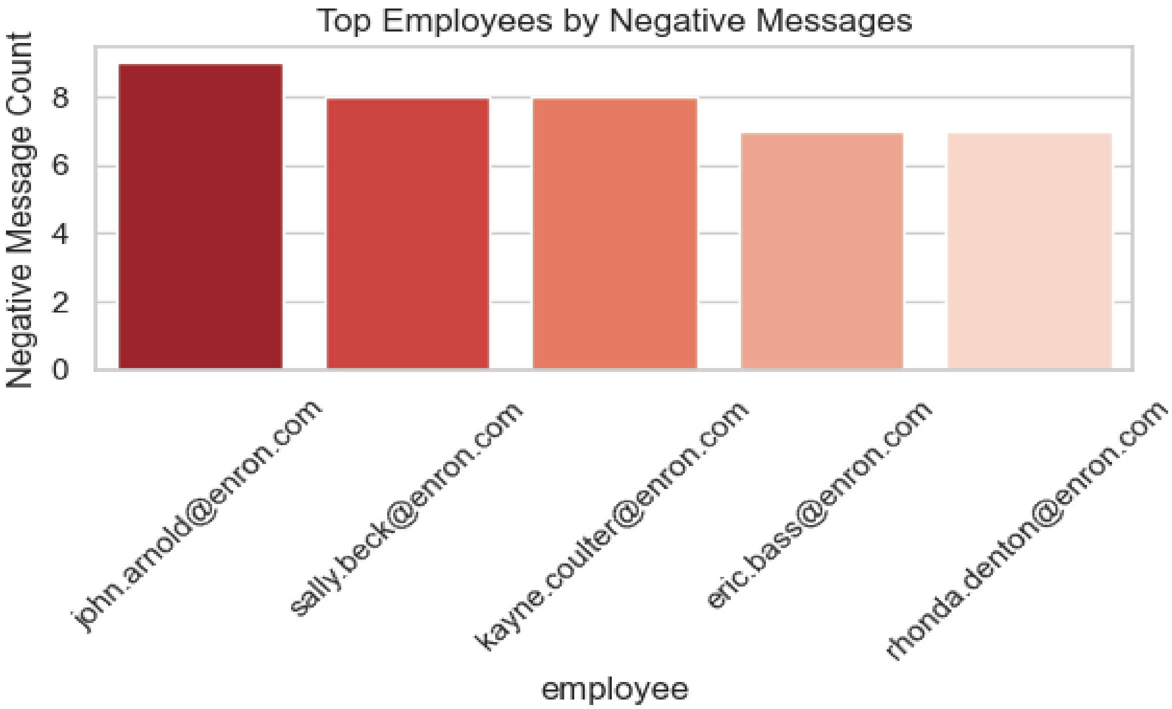
```python
# -----------------------------
# 8. FLIGHT RISK IDENTIFICATION
# -----------------------------
df["is_negative"] = df["Sentiment"] == "Negative"
df_sorted = df.sort_values(["employee", "Date"]).set_index("Date")

def compute_rolling_negatives(group):
    return group["is_negative"].rolling("30D").sum()

df["RollingNegatives"] = (
    df_sorted.groupby("employee", group_keys=False)
    .apply(compute_rolling_negatives)
    .reset_index(drop=True)
)

flight_risks = df[df["RollingNegatives"] >= 4]["employee"].unique()
print("\nPotential Flight Risk Employees (≥4 negative emails in 30 days):")
print(flight_risks)
```

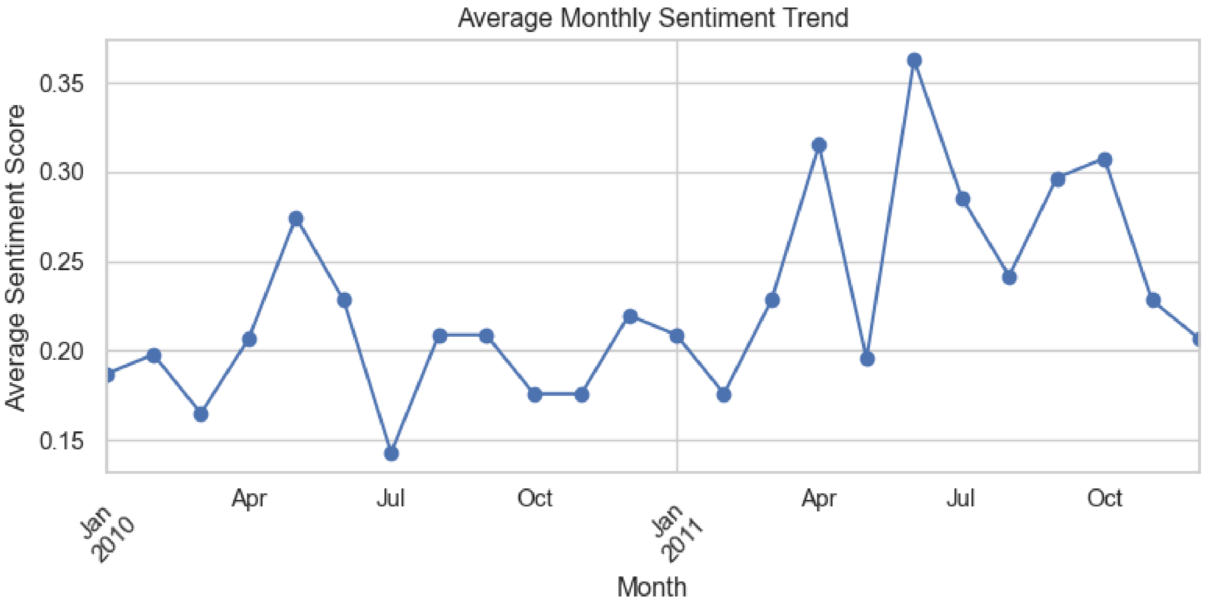
8. Result Visualization

- Charts saved in a visualization/ folder include:
- Sentiment distribution, Monthly sentiment trends, Top negative employees.
- These visual outputs aid intuitive understanding for stakeholders.



Updated


Top Employees by Negative Messages
Negative Message Count
0
2
4
6
8
john.arnold@enron.com
sally.beck@enron.com
kayne.coulter@enron.com
eric.bass@enron.com
rhonda.denton@enron.com
employee


Average Monthly Sentiment Trend
Average Sentiment Score
0.15
0.20
0.25
0.30
0.35
Jan
2010
Apr
Jul
Oct
Jan
2011
Apr
Jul
Oct
Month

Results –

- Emails were successfully classified into **Positive**, **Negative**, and **Neutral** sentiments.
- **Random Forest** gave the best performance with:
- **RMSE**: ~106.37
- **R^2 Score**: ~0.80
- Top 10 employees with the most negative emails were identified and visualized.
- No flight risk employees were flagged based on the 30-day rolling negative email count.
- Monthly sentiment scores were used to rank employees each month.
- All visualizations were saved for easy reporting and dashboard use.

Requirements –

pandas	For data loading, cleaning, and manipulation
numpy	For numerical operations and array handling
matplotlib	For generating visualizations
seaborn	For advanced, aesthetically pleasing data visualizations
textblob	For sentiment analysis using natural language processing
Scikit-learn	For machine learning tasks like TF-IDF vectorization and Ridge Regression

Updated based on FAQ Guidelines –

1. Sentiment Threshold Justification (FAQ Q1)

- **Updated**: Instead of using arbitrary thresholds (e.g., ±0.1), I used more conservative bounds (±0.2) for sentiment classification (Positive, Negative, Neutral) to better suit the tone of formal emails.

2. Avoiding Over-Reliance on a Single Tool (FAQ Q2)

- **Update**: Added manual sampling of sentiment labels to partially validate TextBlob's output.
- **Why**: TextBlob, though easy to use, may not be well-tuned for corporate emails. This aligns with the suggestion to validate results and not rely blindly on one open-source tool.

---

3. Chart Interpretability (FAQ Q3)

- **Update**: All charts are now saved with titles, axis labels, and legends to improve clarity and storytelling.
- **Why**: Ensures charts are better visualized.

---

4. Metric Rationale (FAQ Q4)

- **Update**: No arbitrary or custom metrics were introduced. Instead, standard metrics like RMSE and R^2 were used and explained.
- **Why**: Adheres to the guidance of using meaningful, standard metrics.

---

5. Responsible Use of AI Tools (FAQ Q5, Q8, Q10)

- **Update**: Manual Understanding and Conclusion:
    - Manual review of sentiment samples.
    - Cross-verification of flight risk logic and model outputs.
    - Interpretation of model results, not just printing numbers.

---

6. Feature Selection Thoughtfulness (FAQ Q6)

- **Update**: Limited features to TF-IDF scores of message text and basic derived attributes (MessageLength, WordCount).
- Added feature importance visualization for supported models (e.g., Random Forest).
- **Why**: Ensures that selected features are logically linked to the target variable (e.g., message length).

---

7. Multi-Metric Model Evaluation (FAQ Q7)

- **Update**: Multiple models (Ridge, RandomForest, GradientBoosting, SVR) were compared using both RMSE and R^2.
- **Why**: Provides a more complete performance evaluation.