Part A:

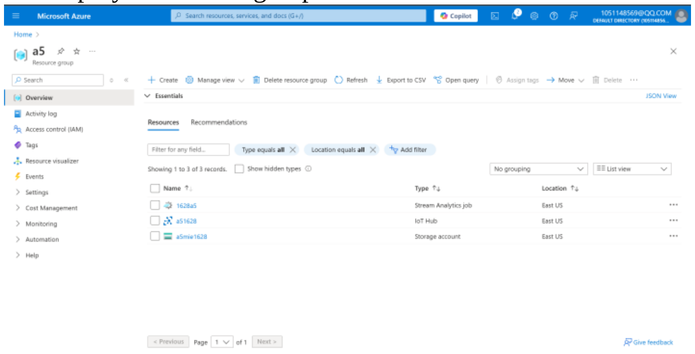
1. Azure Data Lake: A repository for all the raw data (structured, unstructured, etc.) before process and transformation. It should be used as the Data Store in the diagram, because it's flexible in storing the data and cost-effective for a large volume of data.

   Azure Databricks: A managed Spark platform for collaborative data preprocessing, machine learning and analytics. It's great for scalable and distributed processing of large datasets, so it could be used in both Ingest Data and Prepare and Transform Data stages.

   Azure Data Factory: This is a visualized, serverless data integration service for data movement and transformation. It could be used in Ingest Data since it helps move data into data storages. It also fits in Prepare and Transform Data to trigger data transformation jobs.

   Azure Synapse Analytics: A analytics service for fast querying and analysis of large datasets. It fits in the Model and Serve Data part, because it can be used to build data models, perform queries and integrate with tools for visualization, etc.

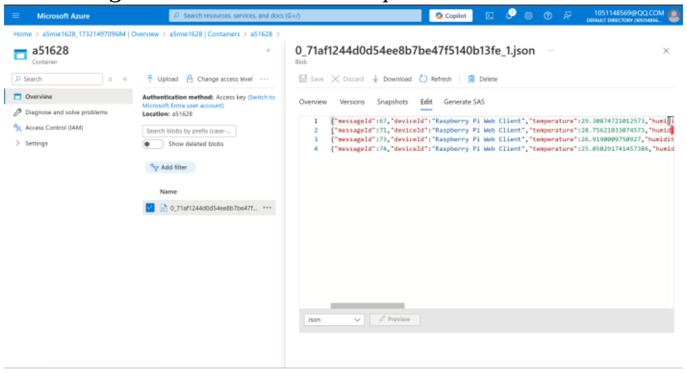
   Azure Cosmos DB: A fully managed NoSQL database. It fits in the Model and Serve Data part, as it's low in latency, highly available and scalable.
2. Stream Analytics identifies patterns and relationships in data. It ingests data from various sources, like Event Hubs and IoT Hub. For processing, it supports SQL-like querying, time-based operations and fast processing. The processed data can be output to destinations, such as Cosmos DB and Blob Storage. Common use cases include fraud detection, where it is used to analyze transactions in real-time for patterns, and IoT monitoring, where it monitors data streams from IoT devices and analyze them for anomalies, optimization.
3. In this question, since my student credit is used up, I used a personal account directly, thus the email being not a U of T email.
   The deployed resource group:



Showing 1 to 3 of 3 records.

Name	Type	Location
1628a5	Stream Analytics job	East US
a51628	IoT Hub	East US
a5mie1628	Storage account	East US

The Storage Account used for this question. Just create a container for the storage account.

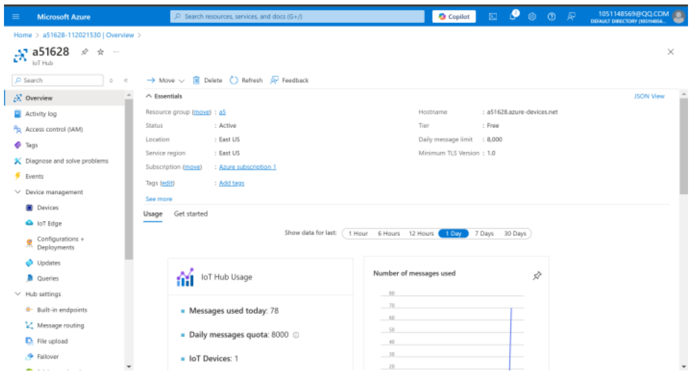


Authentication method: Access key (Switch to Microsoft Entra user account)
Location: a51628

0_71af1244d0d54ee8b7be47f5140b13fe_1.json

```
{"messageId":67,"deviceId":"Raspberry Pi Web Client","temperature":29.30874721012573,"humi
{"messageId":71,"deviceId":"Raspberry Pi Web Client","temperature":28.75621033074573,"humid
{"messageId":73,"deviceId":"Raspberry Pi Web Client","temperature":26.9190009750927,"humidit
{"messageId":74,"deviceId":"Raspberry Pi Web Client","temperature":25.050291741457386,"humid
```

The IoT Hub. Create a device under "Devices".



Home > a51628-112021530 | Overview >

a51628
IoT Hub

Essentials

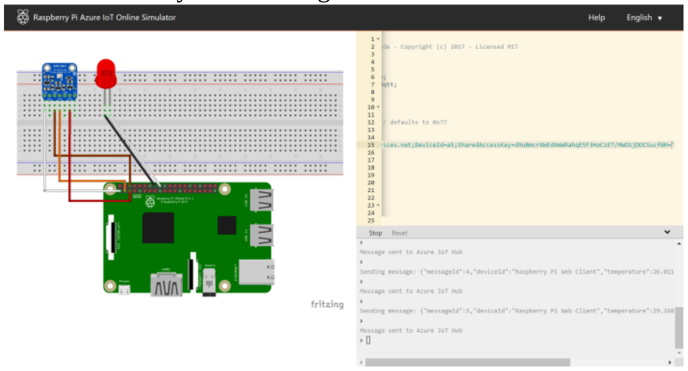
Resource group (move)	: a5	Hostname	: a51628.azure-devices.net
Status	: Active	Tier	: Free
Location	: East US	Daily message limit	: 8,000
Service region	: East US	Minimum TLS Version	: 1.0
Subscription (move)	: Azure subscription 1		
Tags (edit)	: Add tags		

Show data for last: 1 Hour 6 Hours 12 Hours 1 Day 7 Days 30 Days

IoT Hub Usage
- Messages used today: 78
- Daily messages quota: 8000
- IoT Devices: 1

Number of messages used

The RPi simulator. Copy and paste the connections string for the IoT Hub device into line 15. Run and it will continuously send messages to IoT Hub.

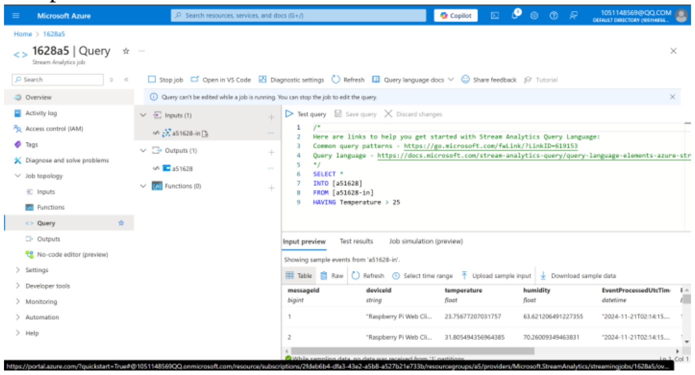


Raspberry Pi Azure IoT Online Simulator

```
15 ices.net;DeviceId=a5;SharedAccessKey=d9uBmcrXWEdXmWRahqE5F3HoCzET/MWO1jDDCSucf0M='
```

```
>
Message sent to Azure IoT Hub
>
Sending message: {"messageId":4,"deviceId":"Raspberry Pi Web Client","temperature":26.011
>
Message sent to Azure IoT Hub
>
Sending message: {"messageId":5,"deviceId":"Raspberry Pi Web Client","temperature":29.268
>
Message sent to Azure IoT Hub
> []
```

The Stream Analytics Job. Specify the input and the output under "Inputs" and "Outputs", then add a query in "Query" as shown below. Notice the entry names are enclosed by square brackets, because it will not compile without them.



Home > 1628a5

1628a5 | Query
Stream Analytics job

Query can't be edited while a job is running. You can stop the job to edit the query.

```
/*
Here are links to help you get started with Stream Analytics Query Language:
Common query patterns - https://go.microsoft.com/fwLink/?LinkID=619153
Query language - https://docs.microsoft.com/stream-analytics-query/query-language-elements-azure-str
*/
SELECT *
INTO [a51628]
FROM [a51628-in]
HAVING Temperature > 25
```

Showing sample events from 'a51628-in'.

messageId *bigint*	deviceId *string*	temperature *float*	humidity *float*	EventProcessedUtcTim *datetime*
1	"Raspberry Pi Web Cli...	23.75677207031757	63.621206491227355	"2024-11-21T02:14:15...
2	"Raspberry Pi Web Cli...	31.805494356964385	70.26009349463831	"2024-11-21T02:14:15...

While sampling data, no data was received from '1' partitions.

https://portal.azure.com/?quickstart=True#@1051148569QQ.onmicrosoft.com/resource/subscriptions/2fdeb6b4-dfa3-43e2-a5b8-a527b21e733b/resourcegroups/a5/providers/Microsoft.StreamAnalytics/streamingjobs/1628a5/ov...

Part B:

1. The Problem Statement:

1a. Description and Objective:

Email spam is still a concern for all of us. Malicious emails could go far beyond ordinary spam into phishing. Automated spam classification could help improve the overall user experience and protect the users. To tackle this problem, the objective of this small project is to develop and deploy an spam classification system on Azure Machine Learning, trained with the Spambase dataset. We expect the system to classify incoming emails as legitimate (ham) / unsolicited (spam), reaching an accuracy of ~95% and false positive of ~5%.

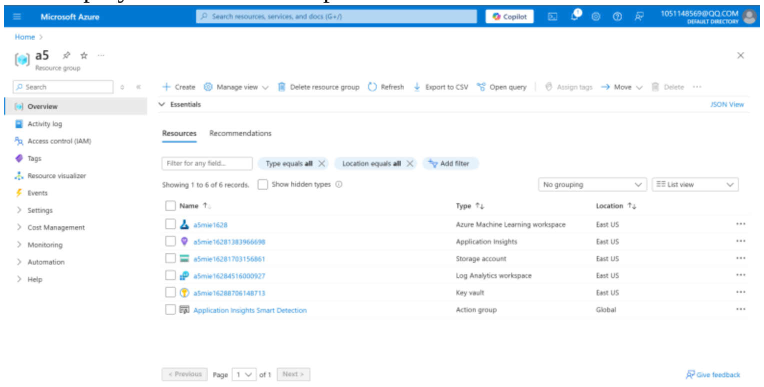
1b. Technical Aspect:

The dataset has 4601 instances with 57 features and 1 binary target variable (spam = 1, ham = 0).

i. Data Preprocessing: In this section we will conduct Data Assessment and Feature Engineering. For the

former, we will verify data completeness and consistency, identify potential correlations between features. For the latter, we will normalize the features based on frequency, aggregate character-based features and identify the most predictive attributes by feature selection.
ii. Data Cleaning: Although the Spambase is preprocessed, we still need to pay attention to feature normalization / standardization for consistent scaling and the potential multicollinearity.
iii. Modeling: The dataset will be split into 70% training set, 15% validation set, 15% test set. And sampling will be stratified to ensure class distribution. For the model, 2 ML models will be used. The model will be evaluated against accuracy, precision, recall, F-1 score, AUC and false positive rate.
For implementation, we will use Azure Notebooks for development and will possibly deploy it in Azure ML.
iv. Timeline: This mini project shall be complete by Nov 30th.

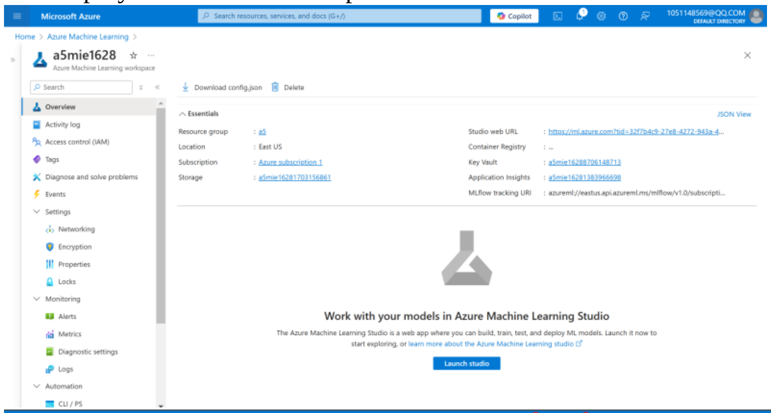
2. The Deployed Resource Group:



Showing 1 to 6 of 6 records.

Name	Type	Location
a5mie1628	Azure Machine Learning workspace	East US
a5mie16281383966698	Application Insights	East US
a5mie16281703156861	Storage account	East US
a5mie16284516000927	Log Analytics workspace	East US
a5mie16288706148713	Key vault	East US
Application Insights Smart Detection	Action group	Global

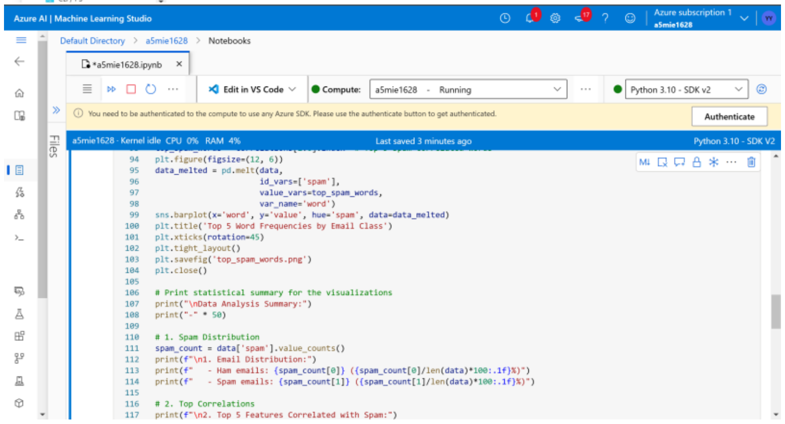
The Deployed Azure ML workspace and Azure Notebook in the Azure Machine Learning Studio:



Resource group	: a5
Location	: East US
Subscription	: Azure subscription 1
Storage	: a5mie16281703156861
Studio web URL	: https://ml.azure.com?tid=32f7b4c9-27e8-4272-943a-4...
Container Registry	: ...
Key Vault	: a5mie16288706148713
Application Insights	: a5mie16281383966698
MLflow tracking URI	: azureml://eastus.api.azureml.ms/mlflow/v1.0/subscripti...

Work with your models in Azure Machine Learning Studio

The Azure Machine Learning Studio is a web app where you can build, train, test, and deploy ML models. Launch it now to start exploring, or learn more about the Azure Machine Learning studio

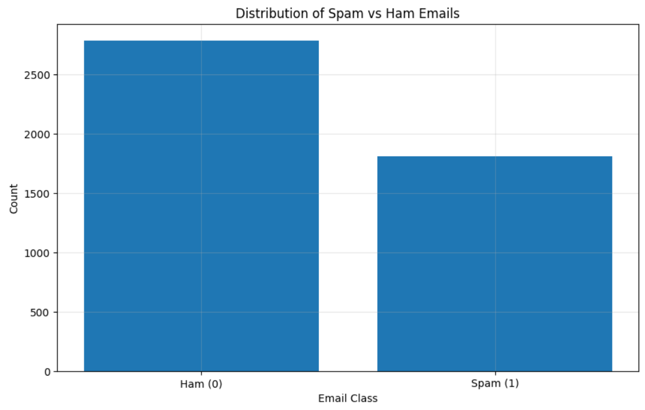

Azure AI | Machine Learning Studio
Azure subscription 1
a5mie1628
Default Directory > a5mie1628 > Notebooks
*a5mie1628.ipynb
Edit in VS Code
Compute: a5mie1628 - Running
Python 3.10 - SDK v2
Authenticate
Files

You need to be authenticated to the compute to use any Azure SDK. Please use the authenticate button to get authenticated.

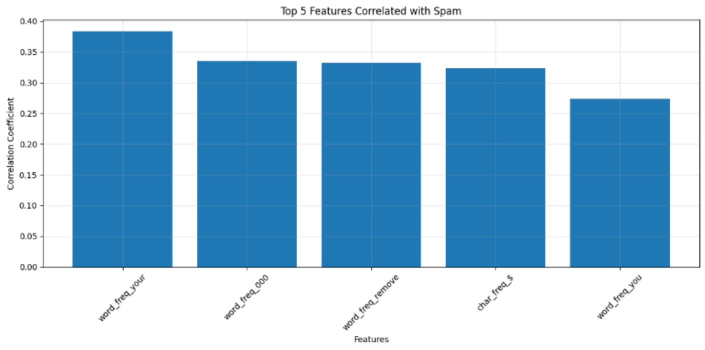
a5mie1628 · Kernel idle CPU 0% RAM 4% Last saved 3 minutes ago Python 3.10 - SDK V2

```python
plt.figure(figsize=(12, 6))
data_melted = pd.melt(data,
                      id_vars=['spam'],
                      value_vars=top_spam_words,
                      var_name='word')
sns.barplot(x='word', y='value', hue='spam', data=data_melted)
plt.title('Top 5 Word Frequencies by Email Class')
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig('top_spam_words.png')
plt.close()

# Print statistical summary for the visualizations
print("\nData Analysis Summary:")
print("-" * 50)

# 1. Spam Distribution
spam_count = data['spam'].value_counts()
print(f"\n1. Email Distribution:")
print(f"   - Ham emails: {spam_count[0]} ({spam_count[0]/len(data)*100:.1f}%)")
print(f"   - Spam emails: {spam_count[1]} ({spam_count[1]/len(data)*100:.1f}%)")

# 2. Top Correlations
print(f"\n2. Top 5 Features Correlated with Spam:")
```

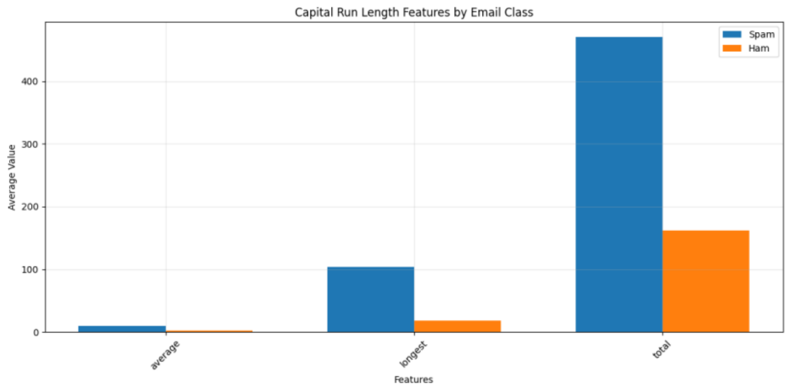
Distribution of Spam vs Ham Emails:
This graph is used to show the class balance in the dataset, which is slightly imbalanced in this case.



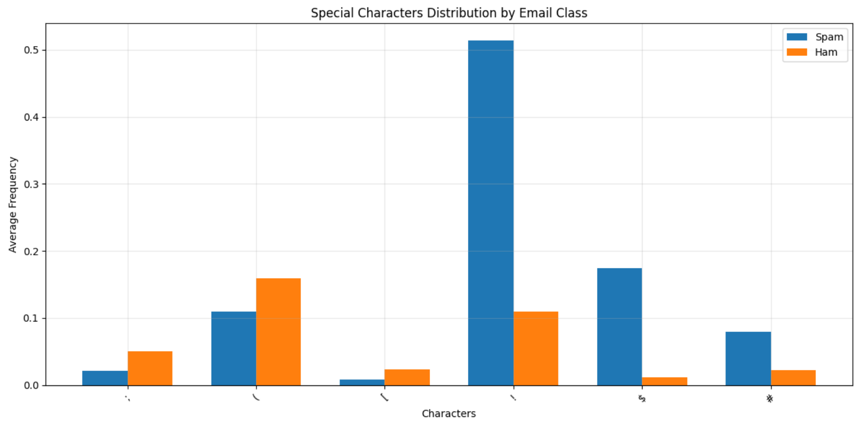
Correlation Heatmap of Top 5 Features:
This graph displays the top 5 features most correlated with spam classification, which might reveal multicollinearity in the model.



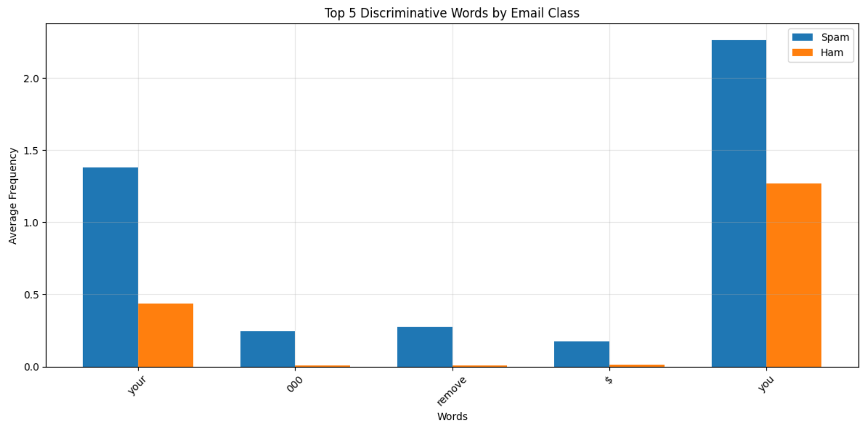
Box Plot of Capital Run Length Features:
This one compares 3 capital letter metrics between spam and ham.



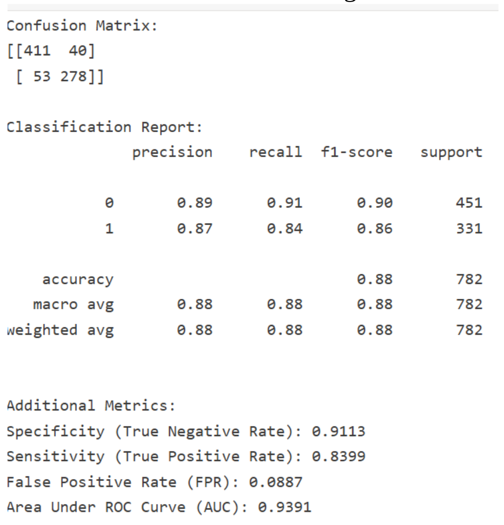
Distribution of Special Characters:
This plot shows the distribution and possibly patterns of special character frequencies.



Word Frequency Analysis for Top 5 Spam-indicator Words:
This is a bar chart comparing the frequencies of the top 5 words indicating a spam.

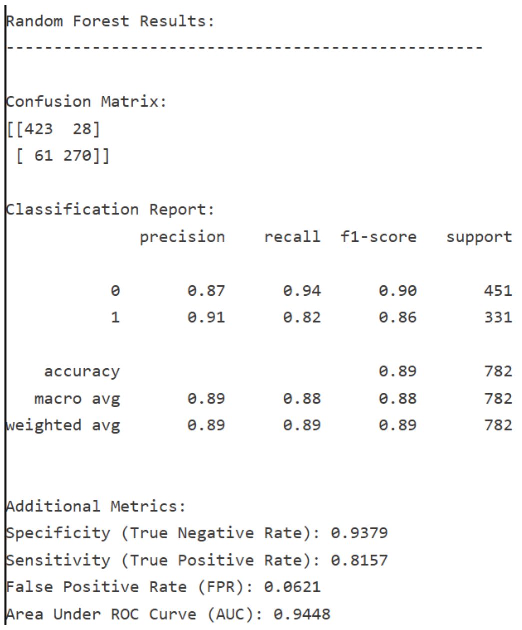


3. Although the dataset is relatively clean, we still need to scale the features, split train/test set with stratification and conduct some basic feature engineering. The features need scaling because different features have vastly different scales. Feature engineering is needed to capture the overall patterns and enhance performance.
4. The models chosen are Random Forest and Gradient Boosting Classification. They were chosen because their internal structure are similar and they are simpler to implement.
   The results of Gradient Boosting:



```
Confusion Matrix:
[[411  40]
 [ 53 278]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.91      0.90       451
           1       0.87      0.84      0.86       331

    accuracy                           0.88       782
   macro avg       0.88      0.88      0.88       782
weighted avg       0.88      0.88      0.88       782


Additional Metrics:
Specificity (True Negative Rate): 0.9113
Sensitivity (True Positive Rate): 0.8399
False Positive Rate (FPR): 0.0887
Area Under ROC Curve (AUC): 0.9391
```

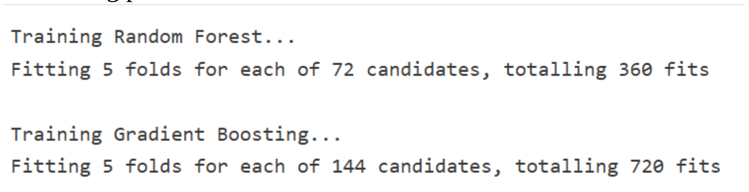
The results of Random Forest:



```
Random Forest Results:
--------------------------------------------------

Confusion Matrix:
[[423  28]
 [ 61 270]]

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.94      0.90       451
           1       0.91      0.82      0.86       331

    accuracy                           0.89       782
   macro avg       0.89      0.88      0.88       782
weighted avg       0.89      0.89      0.89       782


Additional Metrics:
Specificity (True Negative Rate): 0.9379
Sensitivity (True Positive Rate): 0.8157
False Positive Rate (FPR): 0.0621
Area Under ROC Curve (AUC): 0.9448
```

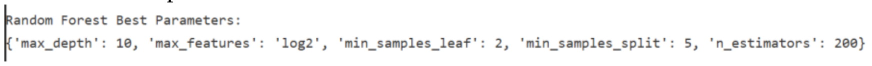
The results showed that the random forest has a slightly better overall performance, but its specificity is relatively lower, indicating it's better at identifying normal emails. However, for gradient boosting, its specificity is closer to sensitivity, meaning that it makes fewer tradeoffs in classification.

5. We did a hyperparameter tuning for the models.
For random forest, the parameters are n_estimators (Number of trees), max_depth (Maximum depth of trees), min_samples_split (Minimum samples required to split), min_samples_leaf (Minimum samples required at leaf node), max_features (Number of features to consider for best split).
For gradient boosting, that would be n_estimators (Number of boosting stages), learning_rate (Contribution of each tree), max_depth (Maximum depth of trees), min_samples_split (Minimum samples required to split), min_samples_leaf (Minimum samples required at leaf node), subsample (Fraction of samples for fitting individual trees).
The tuning process used 5-fold cross-validation, optimized for the AUC and tested multiple combinations of the parameters.
The tuning process:



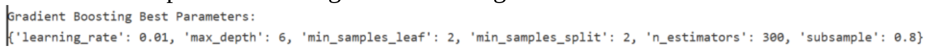
```
Training Random Forest...
Fitting 5 folds for each of 72 candidates, totalling 360 fits

Training Gradient Boosting...
Fitting 5 folds for each of 144 candidates, totalling 720 fits
```

The best set of parameters for random forest:



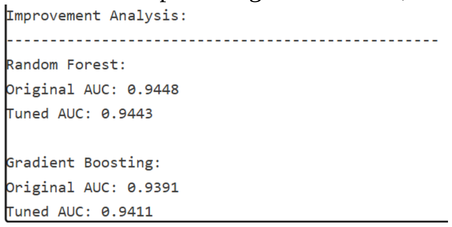
```
Random Forest Best Parameters:
{'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 200}
```

The best set of parameters for gradient boosing:



```
Gradient Boosting Best Parameters:
{'learning_rate': 0.01, 'max_depth': 6, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 300, 'subsample': 0.8}
```

Since we are optimizing for the AUC, this is the improvement after tuning:



```
Improvement Analysis:
--------------------------------------------------
Random Forest:
Original AUC: 0.9448
Tuned AUC: 0.9443

Gradient Boosting:
Original AUC: 0.9391
Tuned AUC: 0.9411
```

6. Conclusions:

Over all the analysis of the results, we conclude that both random forest and gradient boosting are suitable email spam classification, with random forest slightly outperforming with a better AUC (0.9448 compared to 0.9391) The random forest has higher specificity (0.9379) with a lower sensitivity (0.8157), which showed that it's better suited at identifying normal emails and it's more scrupulous in identifying spam emails. Corresponding to its lower false positive rate, this characteristic can be more demanded when false positives are expensive. On the other hand, gradient boosting is more balanced among these 2 metrics, meaning that it's preferred when missing out spam and falsely classifying legitimate emails are of the same importance.

Future work:

The false positive rate could be mitigated by more sophisticated feature engineering techniques. Furthermore, since the difference exists in the models' approaches towards false positives, there might be potentials in ensemble methods that could utilize both models.