Challenges/Issues:

ParserError: Error tokenizing data. C error: EOF inside string starting at row 32437
(116) Python Pandas Error tokenizing data C error EOF inside string starting when reading 1GB CSV file - YouTube
ParserError: Error tokenizing data. C error: EOF inside string starting at row 39665 | by coder for you | Medium

Parsererror is related to issues with the formatting of the file you are trying to read. Some include mismatch columns, unexpected delimiters, unexpected double commas or tabs, etc.

I used these useful links above to understand more of the error.
I learned that the read_csv function has the argument `on_bad_lines,` the which can skip some bad line errors and resume the reading process. Assign the value 'skip' to the argument.

A temporary solution was to just read the file until

pd.read_csv(dataset,on_bad_lines ='skip')

I received the same error again but this time the command was able to read more lines until `253523`:
`ParserError: Error tokenizing data. C error: EOF inside string starting at row 253523`

I looked into stackoverflow and read some of the developer's quick fixes for such an error, python - Pandas ParserError EOF character when reading multiple csv files to HDF5 - Stack Overflow
I learned about another useful argument called `quoting, with the value = csv.QUOTE_NONE`
This parameter indicates that no quotes should be recognized as special characters, meaning the reader will treat quotation marks as regular characters.

Once this parameter was also added, the problem was successfully resolved and the code ran and read the whole csv file without any errors.

```
# Load the normal emails dataset
import csv
normal_emails='/content/emails.csv'
normData=pd.read_csv(normal_emails,on_bad_lines='skip',quoting=csv.QUOTE_NONE)
#df = pd.read_csv(file_path, header = None, delimiter="\t", quoting=csv.QUOTE_NONE, encoding='utf-8')
normData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27277600 entries, 0 to 27277599
Data columns (total 2 columns):
 #   Column     Dtype
---  ------     -----
 0   "file"     object
 1   "message"  object
dtypes: object(2)
memory usage: 416.2+ MB
```

At the end, it turned out that the Kaggle file wasn't fully uploaded and I tried reading it way too early and hence, End of File (EOF) error appeared. However, it was a good exercise to learn about pandas, its functions and parameters.

- Conversion from strings to numbers (One Hot Encoder)

- High computational power and time/ optimization

  When I try visualizing decision trees of the algorithm, the server session expires from excessive use of RAM on Google Collab

  Also, optimization tasks were challenging due to huge running/execution time; I tried with n_estimators(he # of decision trees) with a value of 100, but the run time takes more than 1 hour and sometimes it crashes because of excessive RAM memory. Nonetheless, I had developed a code for Hyperparameter tuning for the best results of the algorithm.

  Results with Decision Tree = 3, execution time: 37 minutes

```
Accuracy: 0.9945992243820038
Precision: 0.9166666666666666
Recall: 0.1647940074906367

precision    recall  f1-score   support

           0       0.99      1.00      1.00     41764
           1       0.92      0.16      0.28       267

    accuracy                           0.99     42031
   macro avg       0.96      0.58      0.64     42031
weighted avg       0.99      0.99      0.99     42031

Accuracy: 0.9945992243820038
Confusion Matrix:
[[41760     4]
 [  223    44]]
```

From the results, it is evident that the model predicts accurately, The precision is high and very good at 91%, although there could be more to improve on.

The model correctly identified 41,760 True Negatives (normal emails), proving that the model is very good at recognizing normal emails.

Only 4 False Positives (normal emails were incorrectly classified as phishing). This indicates a very low rate of false alarms.

223 False Negatives(phishing emails were incorrectly classified as normal). This is a significant number, which indicates that the model is missing a considerable number of phishing emails, which could be problematic since these phishing emails could be a security risk if not identified correctly. This is because only 3 decision trees were used in the test. Generally, Random Forest algorithm performs very well when there are a lot of Decision trees but unfortunately the computation time is very high

The model correctly identified 44 True Positives(phishing emails). Given the total number of phishing emails in the dataset, this number is quite low, reflecting a poor recall.

Next Step:
Optimize the model and tune hyperparameters such as n_estimators and total_depth, use a more powerful GPU for testing.

Appendix: Screenshots of Testing

Train and Test

```
np.random.seed(42)



# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(transformedX, y, test_size=0.2)

themodel = RandomForestClassifier(max_depth=19,n_estimators=369)
# param_dist = {'n_estimators': randint(50,500),
#               'max_depth': randint(1,20)}
# # Use random search to find the best hyperparameters
# rand_search = RandomizedSearchCV(themodel,
#                                  param_distributions = param_dist,
#                                  n_iter=5,
#                                  cv=5)

# # Fit the random search object to the data
# rand_search.fit(X_train, y_train)
themodel.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=19, n_estimators=369)
```

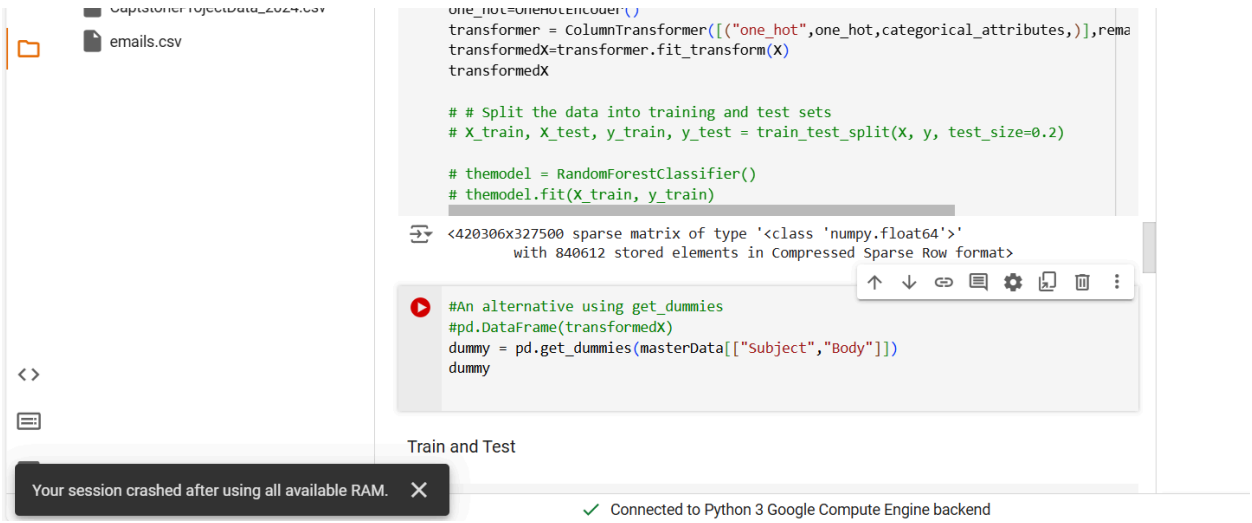
✓ 23m 55s completed at 9:58 PM

```
[ ]  y_pred = themodel.predict(X_test)

     accuracy = accuracy_score(y_test, y_pred)
     precision = precision_score(y_test, y_pred)
     recall = recall_score(y_test, y_pred)

     print("Accuracy:", accuracy)
     print("Precision:", precision)
     print("Recall:", recall)
```

```
Accuracy: 0.9955959844609408
Precision: 0.8888888888888888
Recall: 0.17712177121771217
```



Screenshot of the server crashing after running for 40 min.