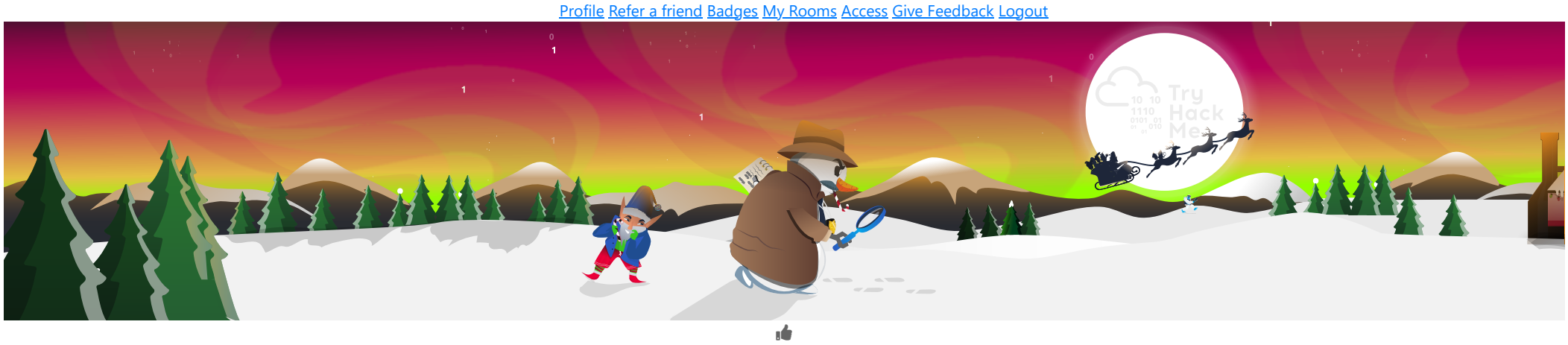


5744

Advent of Cyber 2023

Start AttackBox | Show Split View | Cloud Details | Awards | Help

Get started with Cyber Security in 24 Days - Learn the basics by doing a new, beginner friendly security challenge every day leading up to Christmas.

Chart | Scoreboard | Video | Discuss | Writeups | More

Difficulty: Easy

TryHackMe! Advent of Cyber! • Dec 15, 2023

Source: YouTube

Active Machine Information

Title	IP Address	Expires	
Spam_DetectorLab	10.10.89.77	1h 41m 24s	? Add 1 hour Terminate

100%

Task 1 Introduction Welcome to Advent of Cyber 2023!

Task 2 Introduction Rules and a Short Tutorial

Task 3 Introduction Follow us on social media!

Task 4 Introduction Join the TryHackMe Community!

Task 5 Introduction Subscribing, TryHackMe for Business & Christmas Swag!

Task 6 Introduction The Insider Threat Who Stole Christmas

Task 7 [Day 1] Machine learning Chatbot, tell me, if you're really safe?

Task 8 [Day 2] Log analysis O Data, All Ye Faithful

Task 9 [Day 3] Brute-forcing Hydra is Coming to Town

Task 10 [Day 4] Brute-forcing Baby, it's CeWLd outside

Task 11 [Day 5] Reverse engineering A Christmas DOScovery: Tapes of Yule-tide Past

Task 12 [Day 6] Memory corruption Memories of Christmas Past

Task 13 [Day 7] Log analysis 'Tis the season for log chopping!

Task 14 [Day 8] Disk forensics Have a Holly, Jolly Byte!

Task 15 [Day 9] Malware analysis She sells C# shells by the C2shore

Task 16 [Day 10] SQL injection Inject the Halls with EXEC Queries

Task 17 [Day 11] Active Directory Jingle Bells, Shadow Spells

Task 18 [Day 12] Defence in depth Sleighing Threats, One Layer at a Time

Task 19 [Day 13] Intrusion detection To the Pots, Through the Walls

Task 20 [Day 14] Machine learning The Little Machine That Wanted to Learn

Task 21 [Day 15] Machine learning Jingle Bell SPAM: Machine Learning Saves the Day!

The Story

Start Machine

▶ Click here to watch the walkthrough video!

Over the past few weeks, Best Festival Company employees have been receiving an excessive number of spam emails. These emails are trying to lure users into the trap of clicking on links and providing credentials. Spam emails are somehow ending up in the mailing box. It looks like the spam detector in place since before the merger has been disabled/damaged deliberately. Suspicion is on McGreedy, who is not so happy with the merger.

Problem Statement

McSkidy has been tasked with building a spam email detector using Machine Learning (ML). She has been provided with a sample dataset collected from different sources to train the Machine Learning model.

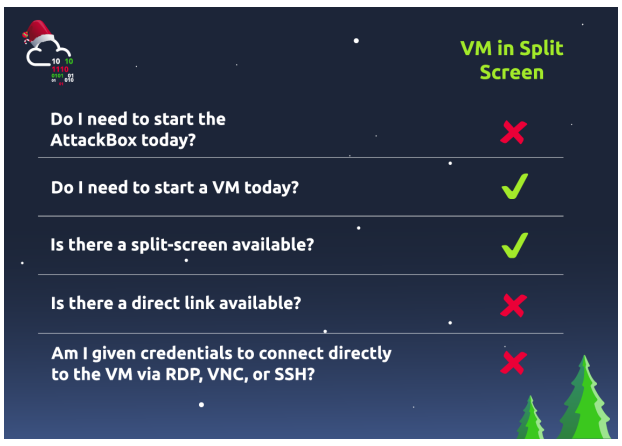
Learning Objectives

In this task, we will explore:

- Different steps in a generic Machine Learning **pipeline**
- Machine Learning classification and training models
- How to split the dataset into training and testing data
- How to prepare the Machine Learning model
- How to evaluate the model's effectiveness

Lab Connection

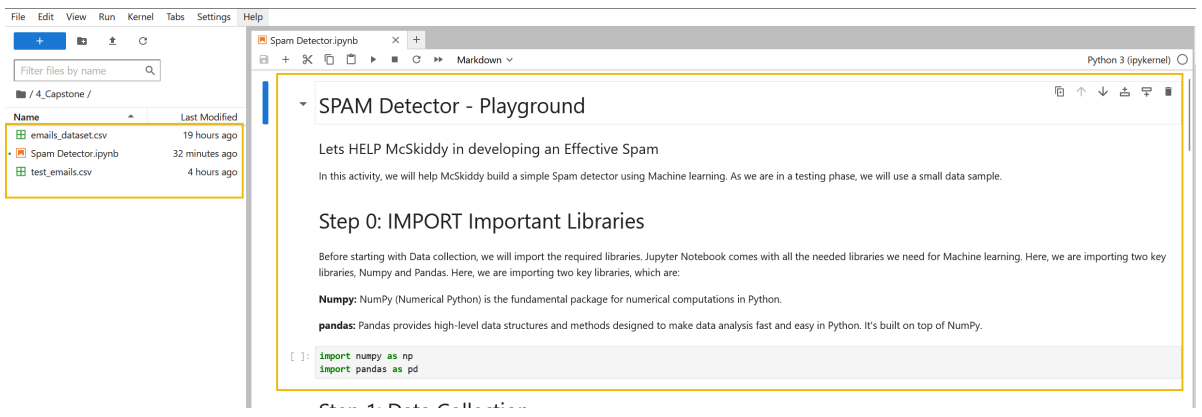
Before moving forward, review the questions in the connection card shown below:



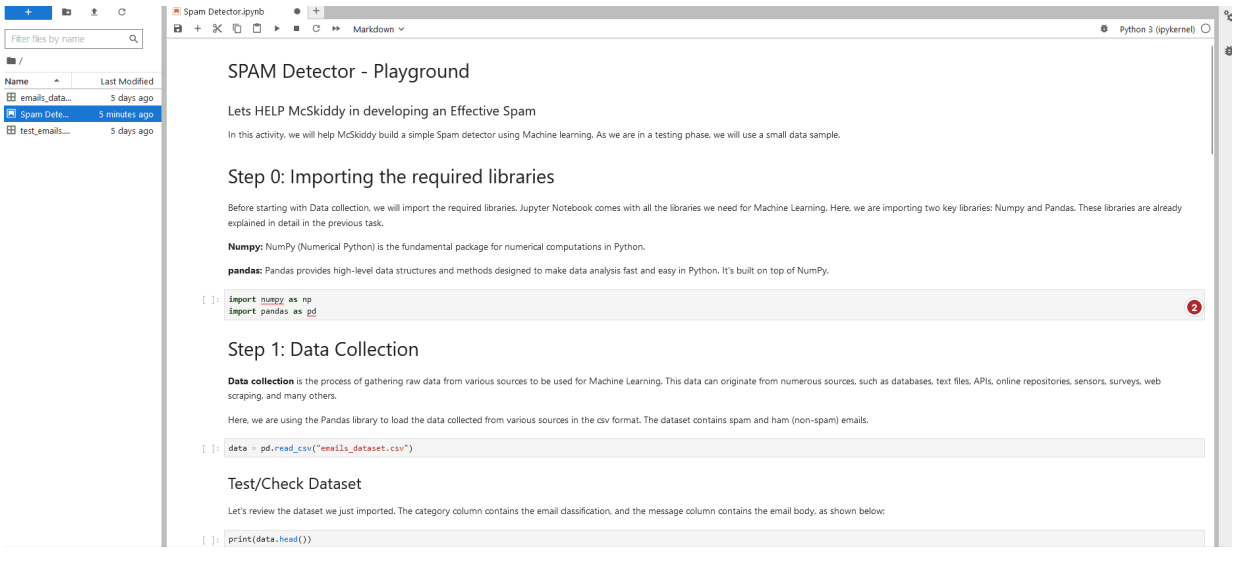
Deploy the machine attached to this task by pressing the green `Start Machine` button at the top-right of this task. After waiting 3-5 minutes, Jupyter will open on the right-hand side. If you cannot see the machine, press the blue "Show Split View" button at the top of the room.

Overview of Jupyter Notebook

Jupyter Notebook provides an environment where you can write and execute code in real time, making it ideal for data analysis, Machine Learning, and scientific research. In this room, we will perform the practical on the Jupyter Notebook.



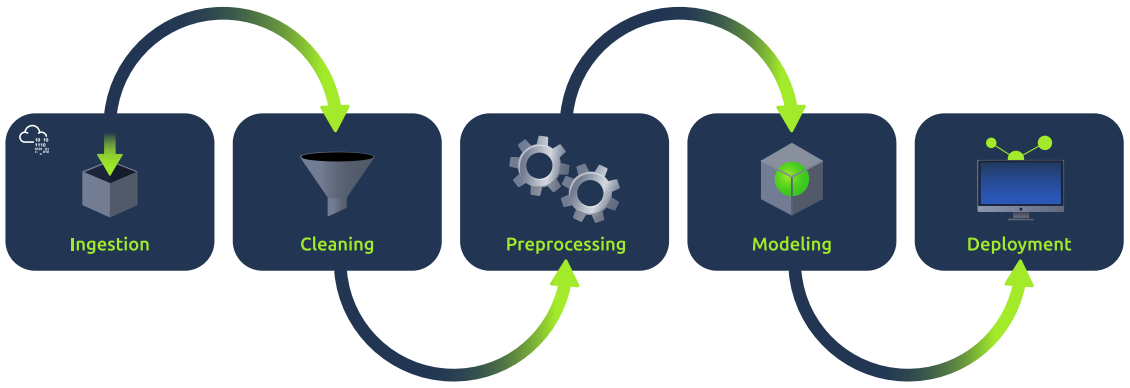
It's important to recall that we will need to run the code from the Cells using the **run button** or by pressing the shortcut `Shift+Enter`. Each step is explained on the Jupyter Notebook for better understanding. Let's dive into the details.



Exploring Machine Learning **Pipeline**

A Machine Learning **pipeline** refers to the series of steps involved in building and deploying an ML model. These steps ensure that data flows efficiently from its raw form to predictions and insights.

A typical **pipeline** would include collecting data from different sources in different forms, preprocessing it and performing feature extraction from the data, splitting the data into testing and training data, and then applying Machine Learning models and predictions.



STEP 0: Importing the required libraries

Before starting with Data collection, we will import the required libraries. Jupyter Notebook comes with all the libraries we need for Machine Learning. Here, we are importing two key libraries: Numpy and Pandas. These libraries are already explained in detail in the previous task.

```
 import numpy as np
import pandas as pd
```

Let's start our SPAM EMAIL detection in the following steps:

Step 1: Data Collection

Data collection is the process of gathering raw data from various sources to be used for Machine Learning. This data can originate from numerous sources, such as databases, text files, APIs, online repositories, sensors, surveys, web scraping, and many others.

Here, we are using the Pandas library to load the data collected from various sources in the csv format. The dataset contains spam and ham (non-spam) emails.

```
data = pd.read_csv("emails_dataset.csv")
```

Test/Check Dataset

Let's review the dataset we just imported. The category column contains the email classification, and the message column contains the email body, as shown below:

```
print(data.head())
```

Expected Output

```
 Classification                                            Message
0           spam  Congratulations !! You have won the Free ticket
1            ham  Call me back when you get the message.
2            ham  Nah I don't think he goes to usf, he lives aro...
3           spam  FreeMsg Hey there darling it's been 3 week's n...
4            ham  Even my brother is not like to speak with me. ... ...
```

DataFrames provide a structured and tabular representation of data that's intuitive and easy to read. Using the command below, let's use the pandas library to convert the data into a frame. It will make the data easy to analyse and manipulate.

```
 df = pd.DataFrame(data)
print(df)
```

Expected Output

```
 Classification                                                Message
0             spam Congratulations !! You have won the Free ticket 1 ham Call me back when you get
the message. 2 ham Nah I don't think he goes to usf, he lives aro... 3 spam FreeMsg Hey there
darling it's been 3 week's n... 4 ham Even my brother is not like to speak with me. ... ...
...                                                         ...
5565          spam  This is the 2nd time we have tried 2 contact u...
5566           ham               Will ü b going to esplanade fr home?
5568           ham You have Won the Ticket Lottery
5569           ham funny as it sounds. Its true to its name

[5570 rows x 2 columns]
```

Step 2: Data Preprocessing

Data preprocessing refers to the techniques used to convert raw data into a clean, organised, understandable, and structured format suitable for Machine Learning. Given that raw data is often messy, inconsistent, and incomplete, preprocessing is an essential step to ensure that the data feeding into the ML models is relevant and of high quality. Here are some common techniques used in data preprocessing:

Technique	Description	Use Cases
Cleaning	Correct errors, fill missing values, smooth noise, and handle outliers.	To ensure the quality and consistency of the data.
Normalization	Scaling numeric data into a uniform range, typically [0, 1] or [-1, 1].	When features have different scales and we want equal contribution from all features.
Standardization	Rescaling data to have a mean (μ) of 0 and a standard deviation (σ) of 1 (unit variance).	When we want to ensure that the variance is uniform across all features.
Feature Extraction	Transforming arbitrary data such as text or images into numerical features.	To reduce the dimensionality of data and make patterns more apparent to learning algorithms.
Dimensionality Reduction	Reducing the number of variables under consideration by obtaining a set of principal variables.	To reduce the computational cost and improve the model's performance by reducing noise.
Discretization	Transforming continuous variables into discrete ones.	To handle continuous variables and make the model more interpretable.
Text Preprocessing	Tokenization, stemming, lemmatization, etc., to convert text to a format usable for ML algorithms.	To process and structure text data before feeding it into text analysis models.
Imputation	Replacing missing values with statistical values such as mean, median, mode, or a constant.	To handle missing data and maintain the dataset's integrity.
Feature Engineering	Creating new features or modifying existing ones to improve model performance.	To enhance the predictive power of the learning algorithms by creating features that capture more information.

Utilizing CountVectorizer()

Machine Learning models understand numbers, not text. This means the text needs to be transformed into a numerical format. `CountVectorizer`, a class provided by the `scikit-learn` library in Python, achieves this by converting text into a token (word) count matrix. It is used to prepare the data for the Machine Learning models to use and predict decisions on.

Here, we are using the `CountVectorizer` function from the sklearn library.

```
 from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(df['Message'])
```

```
print(X)
```

Expected Output

```
  (0, 77)   1
 (0, 401)  1
 (0, 410)  1
 (0, 791)  1
 (0, 1165) 1
 (0, 2173) 1
 (0, 2393) 1
 (0, 2958) 2
 (0, 3095) 2
 (0, 3216) 1
 (0, 3368) 1
 .......
 ......
 ......
```

Step 3: Train/Test Split dataset

It's important to test the model's performance on unseen data. By splitting the data, we can train our model on one subset and test its performance on another.

Here, variable X contains the dataset. We will use the functions from the sklearn library to split the dataset into training data and testing data, as shown below:

```
 from sklearn.model_selection import train_test_split
y = df['Classification']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

- **X**: The first argument to `train_test_split` is the feature matrix `X` which you obtained from the `CountVectorizer`. This matrix contains the token counts for each message in the dataset.
- **y**: The second argument is the labels for each instance in your dataset, which indicates whether a message is spam or ham.
- **test_size=0.2**: This argument specifies that 20% of the dataset should be kept as the test set and the rest (80%) should be used for training. It's a common practice to hold out a portion of the dataset for testing to evaluate the performance of the model on unseen data. This is where the actual splitting of data into training and test sets happens.

The function then returns four values:

- **X_train**: The subset of the features to be used for training.
- **X_test**: The subset of the features to be used for testing.
- **y_train**: The corresponding labels for the X_train set.
- **y_test**: The corresponding labels for the X_test set.

Step 4: Model Training

Now that we have the dataset ready, the next step would be to choose the text classification model and use it to train on the given dataset. Some commonly used text classification models are explained below:

Model	Explanation
Naive Bayes Classifier	A probabilistic classifier based on Bayes' Theorem with an assumption of independence between features. It's particularly suited for high-dimensional text data.
Support Vector Machine (SVM)	A robust classifier that finds the optimal hyperplane to separate different classes in the feature space. Works well with non-linear and high-dimensional data when used with kernel functions.
Logistic Regression	A statistical model that uses a logistic function to model a binary dependent variable, in this case, spam or ham.
Decision Trees	A model that uses a tree-like graph of decisions and their possible consequences; it's simple to understand but can overfit if not pruned properly.
Random Forest	An ensemble of decision trees, typically trained with the "bagging" method to improve the predictive accuracy and control overfitting.
Gradient Boosting Machines (GBMs)	An ensemble learning method is building strong predictive models in a stage-wise fashion; known for outperforming random forests if tuned correctly.
K-Nearest Neighbors (KNN)	A non-parametric method that classifies each data point based on the majority vote of its neighbors, with the data point being assigned to the class most common among its k nearest neighbors.

Model Training using Naive Bayes

Naive Bayes is a statistical method that uses the probability of certain words appearing in spam and non-spam emails to determine whether a new email is spam or not.

How Naive Bayes Classification Works

- Let's say we have a bunch of emails, some labelled as "spam" and others as "ham".
- The Naive Bayes algorithm learns from these emails. It looks at the words in each email and calculates how frequently each word appears in spam or ham emails. For instance, words like "free", "win", "offer", and "lottery" might appear more in spam emails.
- The Naive Bayes algorithm calculates the probability of the email being spam based on the words it contains.
- When the model is trained with Naive Bayes and gets a new email that says (for example) "Win a free toy now!", then it thinks:
  - "Win" often appears in spam, so this increases the chance of the email being spam.
  - "Free" is also common in spam, further increasing the spam probability.
  - "Toy" might be neutral, often appearing in both spam and ham.
  - After considering all the words, it calculates the overall probability of the email being spam and ham.

If the calculated probability of spam is higher than that of ham, the algorithm classifies the email as spam. Otherwise, it's classified as ham.

Let's use Naive Bayes to train the model, as shown and explained below:

```
 from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(X_train, y_train)
```

- **X_train:** This is the training data you want the model to learn from. It's the token counts for each message in the training dataset, obtained from the CountVectorizer.
- **y_train:** These are the correct labels (either "spam" or "ham") for each message in the X_train dataset.

This is where the actual training of the model happens. The fit method is used to train or "fit" the model on your training data.

When we call the `fit` method, the `MultinomialNB` model goes through the data and learns patterns. In the context of Naive Bayes, it calculates the probabilities and likelihoods of each feature (word/token) being associated with each class (spam/ham). These calculations are based on Bayes' theorem and the assumption of feature independence given the class label.

Once the model has been trained with the `fit` method, it can be used to make predictions on new, unseen data.

Step 5: Model Evaluation

After training, it's essential to evaluate the model's performance on the test set to gauge its predictive power. This will give you metrics such as accuracy, precision, and recall.

```
 from sklearn.metrics import classification_report
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

Expected Output

```
              precision    recall  f1-score   support

         ham       0.99      0.99      0.99       957
        spam       0.94      0.96      0.95       157

    accuracy                           0.98      1114
   macro avg       0.97      0.97      0.97      1114
weighted avg       0.98      0.98      0.98      1114
```

The classification_report function takes in the true labels (y_test) and the predicted labels (y_pred) and returns a text report showing the main classification metrics.

- **Precision:** This is the ratio of correctly predicted positive observations to the total predicted positives. The question it answers is: Of all the samples predicted as positive, how many were actually positive?
- **Recall (sensitivity):** The ratio of correctly predicted positive observations to all the actual positives. It answers the question: Of all the actual positive samples, how many did we predict correctly?
- **F1-score:** The harmonic mean of the precision and recall metrics. It gives a better measure of the incorrectly classified cases than the accuracy metric, especially when there's an imbalance between classes.
- **Support:** This metric is the number of actual occurrences of the class in the specified dataset.
- **Accuracy:** The ratio of correctly predicted observations to the total observations.
- **Macro Avg:** This averages the unweighted mean per label.
- **Weighted Avg:** This metric averages the support-weighted mean per label.

The report gives us insights into how well your model is performing for each class and overall, in terms of these metrics.

Step 6: Testing the Model

Once satisfied with the model's performance, we can use it to classify new messages and determine if they are spam or ham.

```
 message = vectorizer.transform(["Today's Offer! Claim ur £150 worth of discount vouchers! Text YES to 85023 now! SavaMob, member offers mobile! T Cs 08717898035. £3.00 Sub. 16 . Unsub reply X "])
prediction = clf.predict(message)
print("The email is :", prediction[0])
```

What's Next?

McSkidy is happy that a workable SPAM detector model has been developed. She has provided us with some test emails in the file `test_emails.csv` and wants us to run the prepared model against these emails to test our model results.

```
 test_data = pd.read_csv("______")
print(test_data.head())
```

Expected Output

```
                                          Messages
0  Reply with your name and address and YOU WILL ...
1  Kind of. Took it to garage. Centre part of exh...
2                    Fighting with the world is easy
3  Why must we sit around and wait for summer day...
```

```
 X_new = vectorizer.transform(test_data['Messages'])
new_predictions = clf.predict(X_new)
results_df = pd.DataFrame({'Messages': test_data['Messages'], 'Prediction': new_predictions})
print(results_df)
```

Expected Output

```
  Messages                                                Prediction
0   Reply with your name and address and YOU WILL ...       spam
1   Kind of. Took it to garage. Centre part of exh...        ham
2                     Fighting with the world is easy        ham
3   Why must we sit around and wait for summer day...        ham
----------REDACTED OUTPUT---------------------------------
```

Conclusion

This is it from the task. From the practical point of view, we have to consider the following points to ensure the effectiveness and reliability of the model:

- Continuously monitor the model's performance on a test dataset or in a real-world environment.
- Collect feedback from end-users regarding false positives.
- Use this feedback to understand the model's weaknesses and areas for improvement.
- Deploy the model into production.

Answer the questions below

What is the key first step in the Machine Learning **pipeline**?

data collection — Correct Answer

Which data preprocessing feature is used to create new features or modify existing ones to improve model performance?

feature engineering — Correct Answer

During the data splitting step, 20% of the dataset was split for testing. What is the percentage weightage avg of precision of spam detection?

0.98 — Correct Answer — Hint

How many of the test emails are marked as spam?

3 — Correct Answer

One of the emails that is detected as spam contains a secret code. What is the code?

I_hate_best_festival — Correct Answer

If you enjoyed this room, please check out the Phishing module.

No answer needed — Correct Answer

Created by 

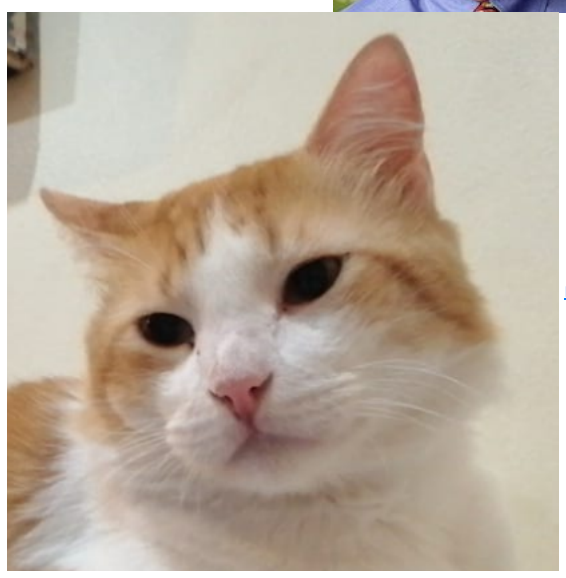
tryhackme and


ar33zy and


cmnatic and


Dex01 and

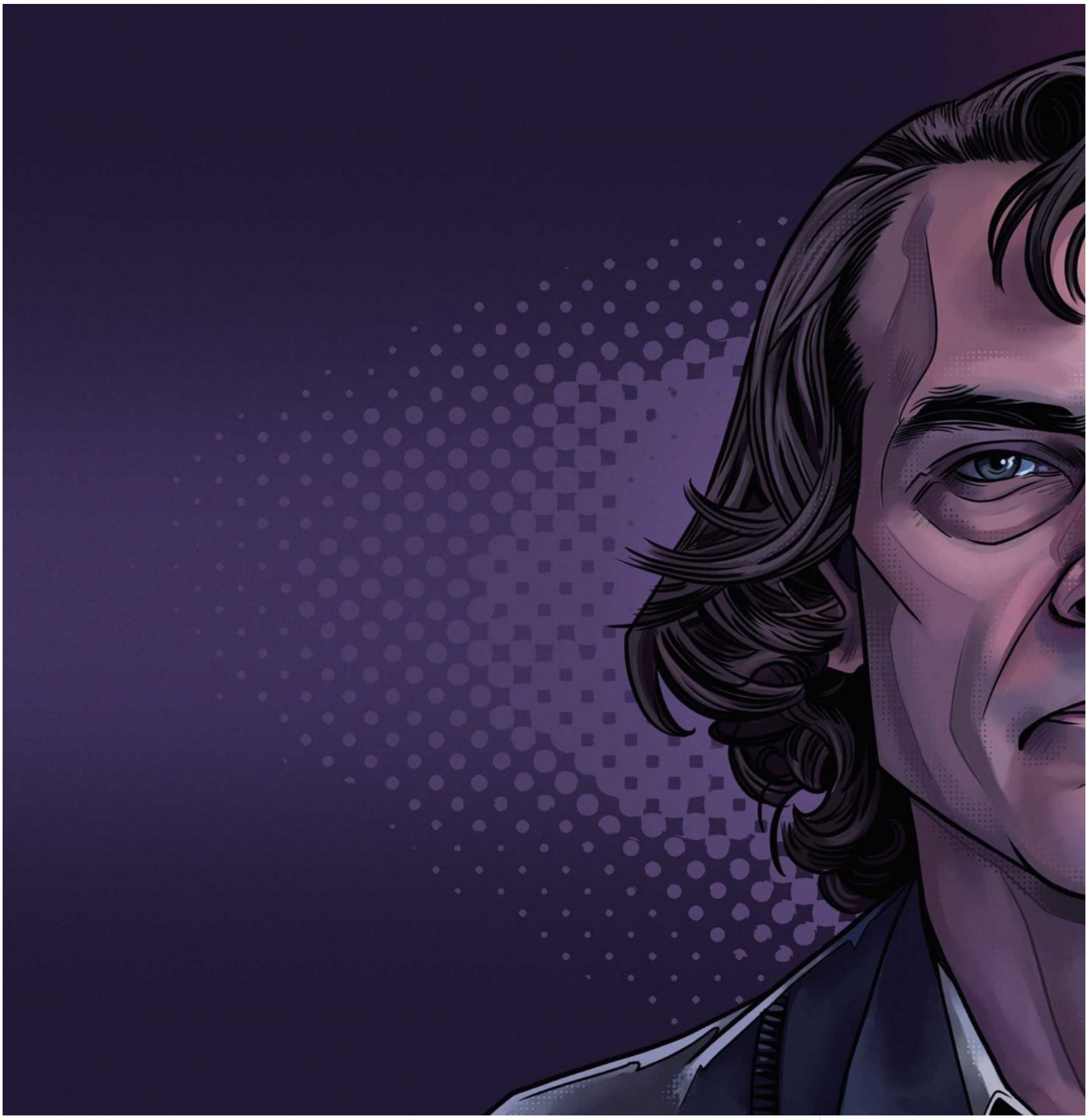

timtaylor and


munra and

hk and

strategos and

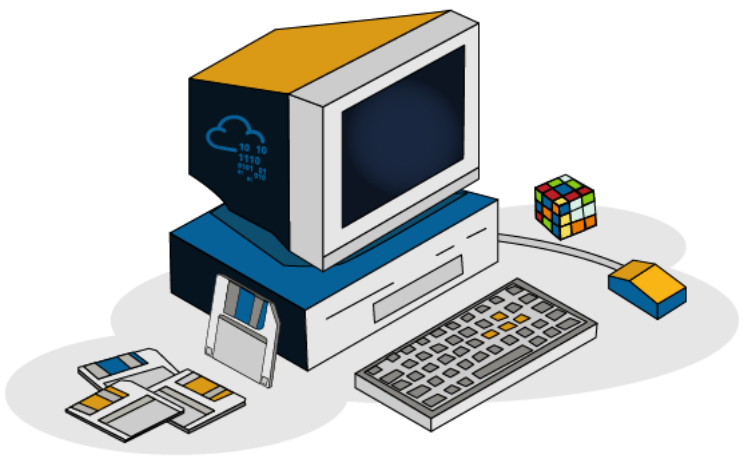
Fontaene and



[SecurityNomad](#) and

[am03bam4n](#) and

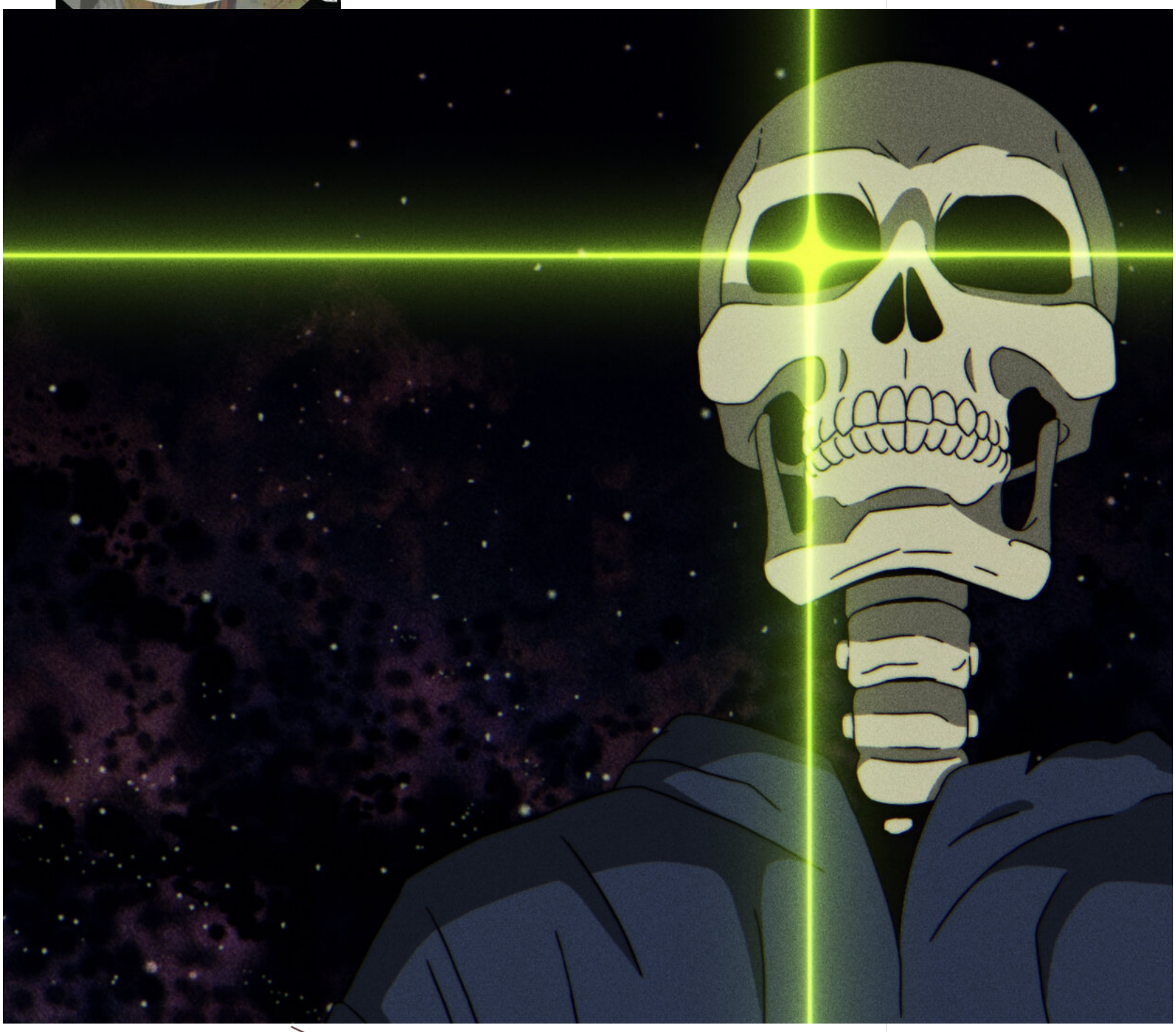
[Mokmokmok](#) and

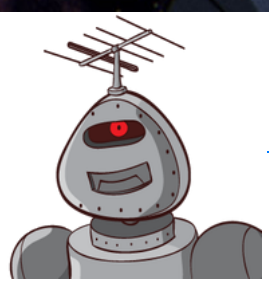
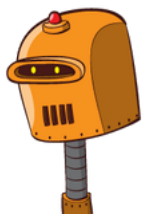


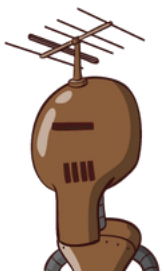
[umairalizafar](#) and

[ujohn](#)

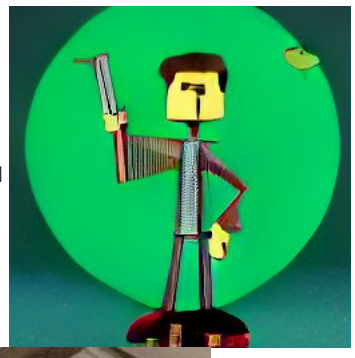
and  hadrian3689 and  melmols and



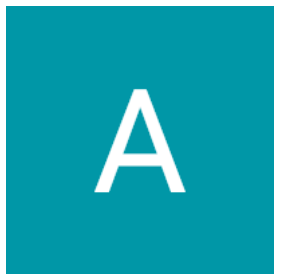
MaxRobertson and  1337rce and  MartaStrzelec and



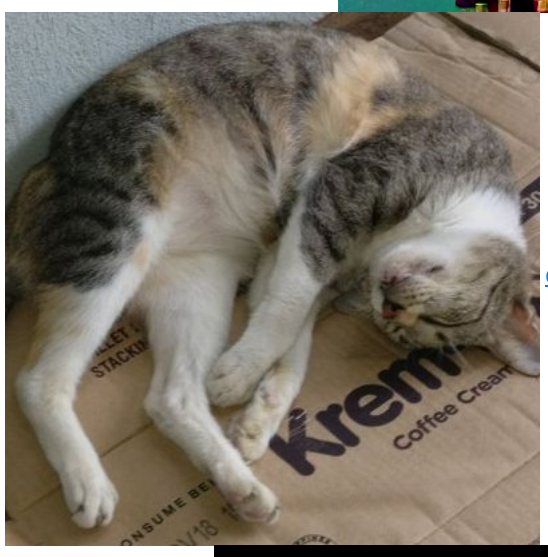
[muhammadzaryab](#) and



[arebel](#) and

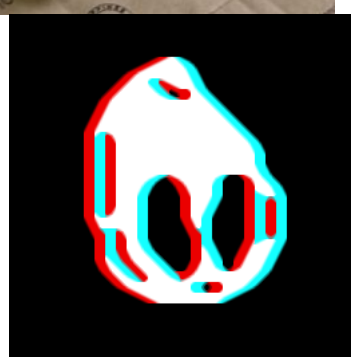


[andrea526](#) and



[Orzykf](#) and

[l000g1c](#) and



[odacavoTHM](#)

This is a **free** room, which means anyone can deploy virtual machines in the room (without being subscribed)! 69351 users are in here and this room is 9 days old.