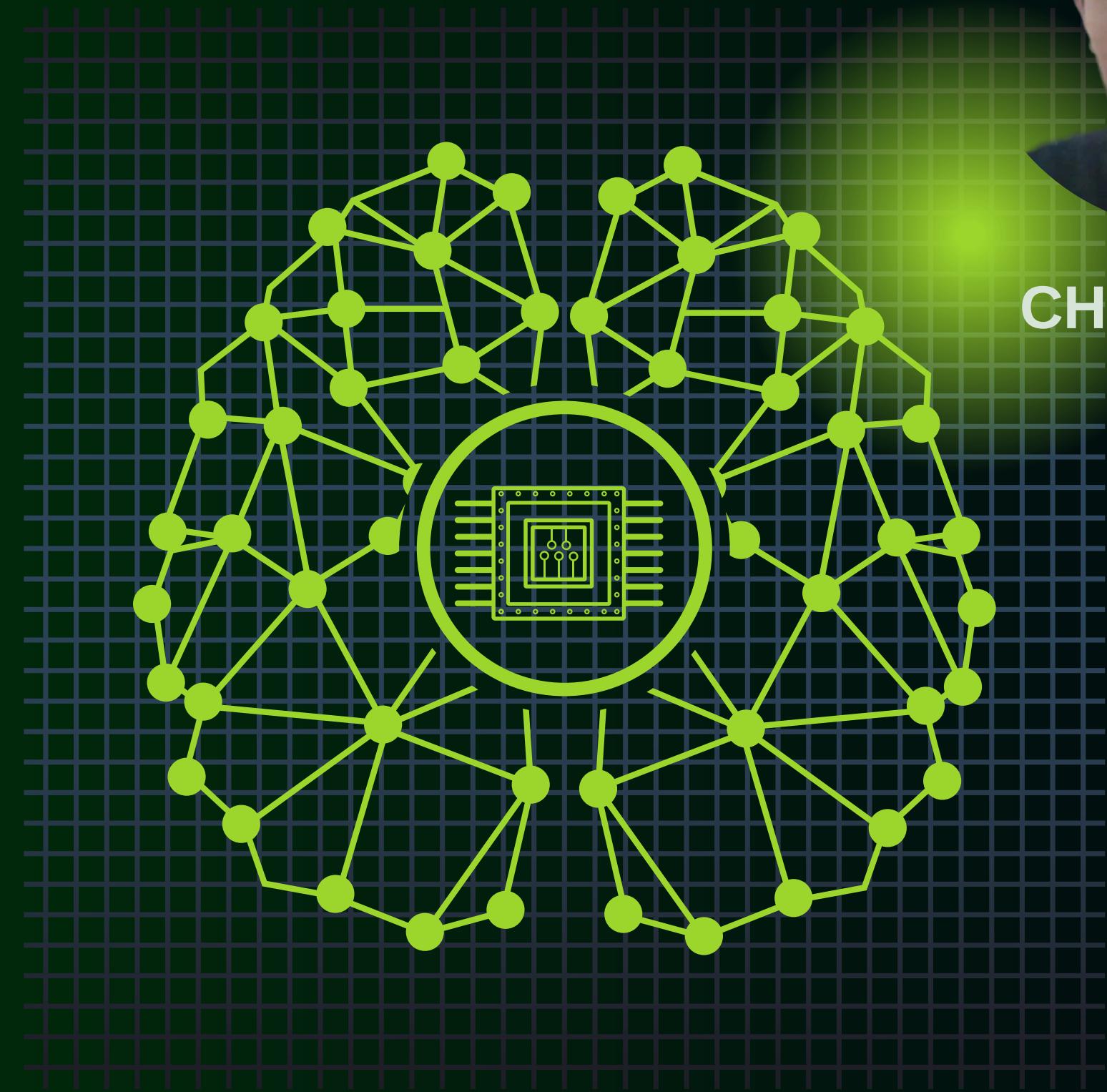


GROUP 39

# GAI AND MACHINE LEARNING



CHONG GUI



# LEARNING/ASSESSMENT TASK

To predict whether a passenger would recommend an airline based on passenger service factors and how this differed before and after covid using a Support Vector Machine.



CHONG GUI

## 01 Data Preprocessing

- a. Converting flight dates into datetime object
- b. Splitting the data into pre and post COVID
- c. Removing irrelevant noise

## 02 Handling Missing Values

- a. Impute missing values

## 03 Balancing dataset

- a. Generate synthetic data to ensure fair representation of data from both periods

## 04 Model Training

- a. Hyperparameter optimisation

# LEARNING/ASSESSMENT TASK

To predict whether a passenger would recommend an airline based on passenger service factors and how this differed before and after covid using a Support Vector Machine.



CHONG GUI

## 05 Model Evaluation

- a. Possible Performance Metrics
  - a.i. Accuracy
  - a.ii. Precision
  - a.iii. Recall
  - a.iv. F1-score
- b. Data Visualisations
  - b.i. ROC Curve
  - b.ii. Area Under Curve (AUC)

# Without GAI Tools:



SU TONG

Data Processing → Date conversion

Missing values

Data splitting

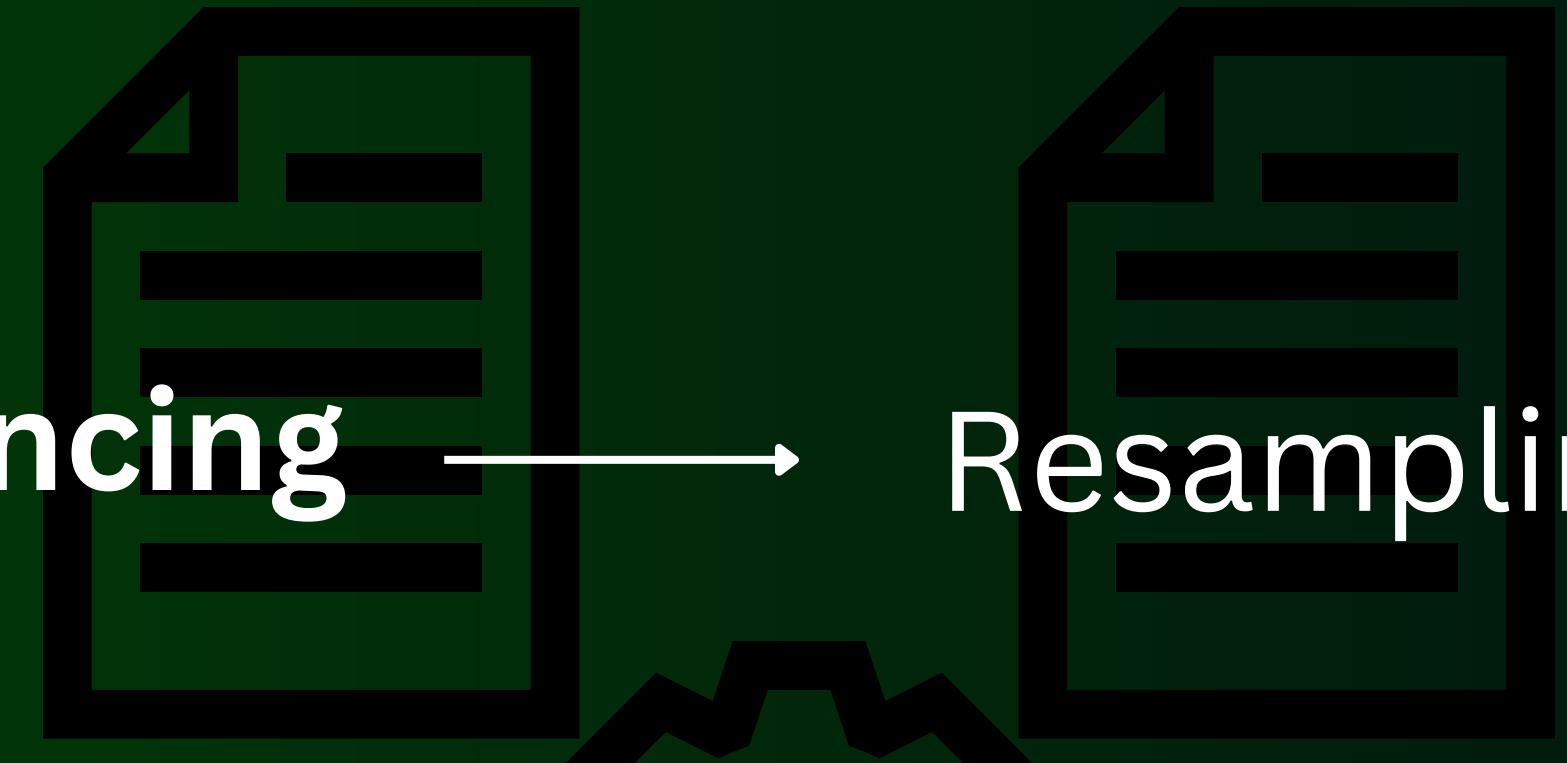
Removing irrelevant noise

# Without GAI Tools:



SU TONG

Data Balancing



Resampling techniques

# Without GAI Tools:

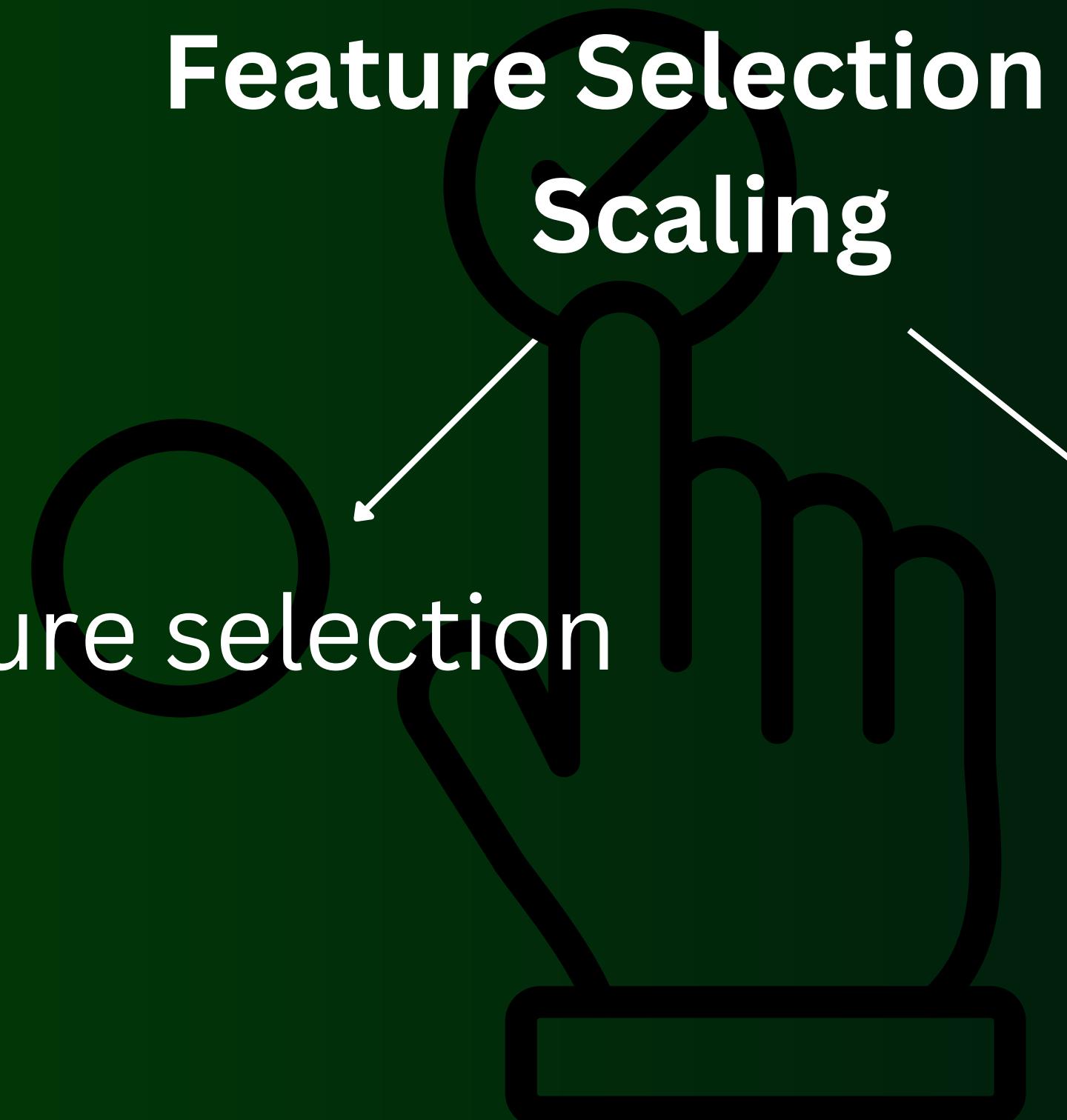


SU TONG

Feature Selection and  
Scaling

Manual feature selection

Feature scaling



# Without GAI Tools:

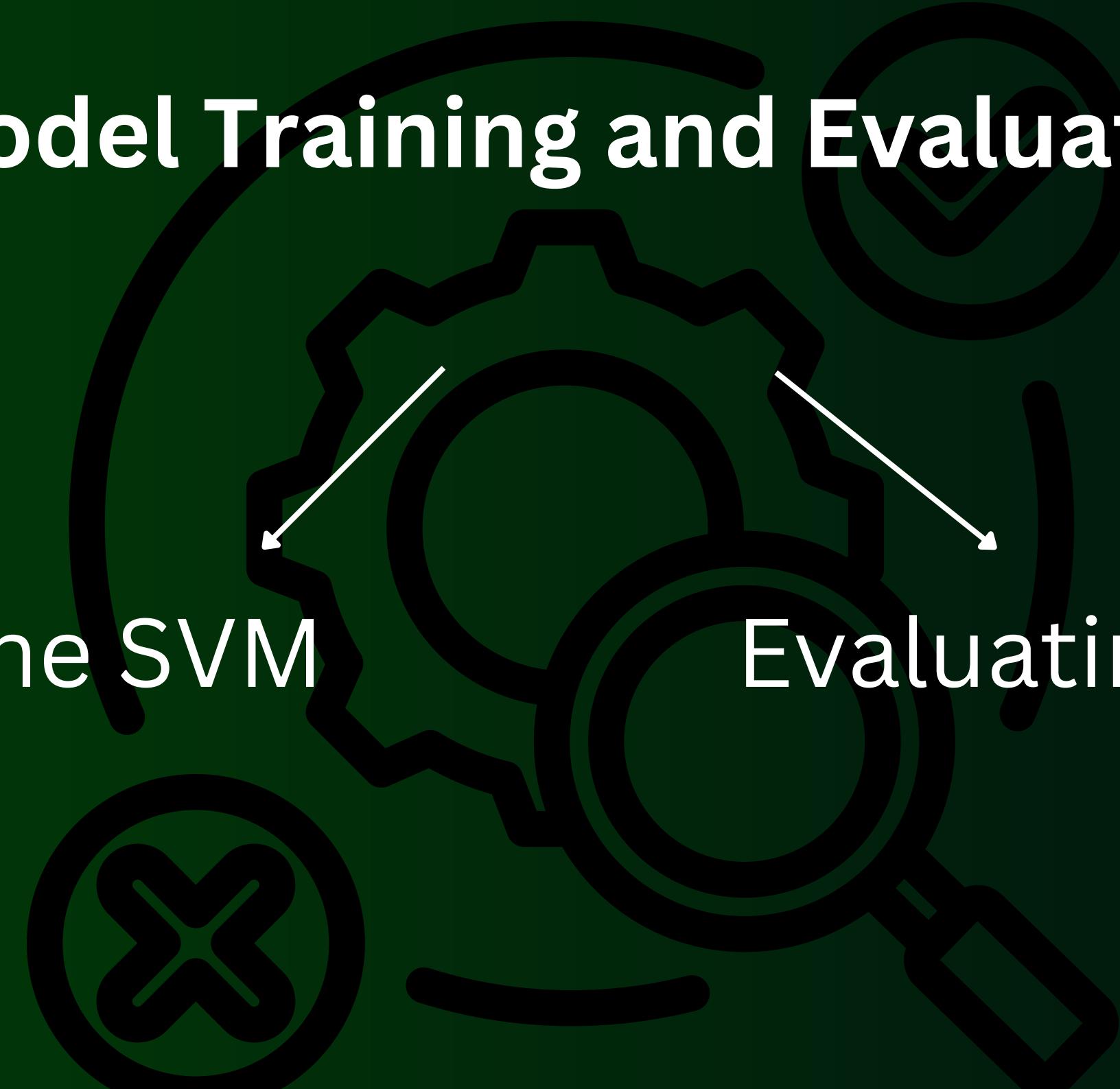
Model Training and Evaluation



SU TONG

Training the SVM

Evaluating performance



# Without GAI Tools:



Critical Analysis



SU TONG



JIE QI

IMPROVE THE LEARNING OR  
ASSESSMENT PROCESS WITH  
THE USE OF GAI TOOLS.

# EFFICIENT DATA PREPROCESSING



JIE QI

There are special characters and unnecessary whitespaces in the dataset.  
Generate code to remove this unnecessary noise from the dataset

↔ Data Augmentation Reviews

```
1 import pandas as pd
2 import numpy as np
3 import random
4 import re
5 import matplotlib.pyplot as plt
6 import seaborn as sns
7 from sklearn.model_selection import train_test_split, GridSearchCV
8 from sklearn.preprocessing import LabelEncoder, StandardScaler
9 from sklearn.svm import SVC
10 from sklearn.metrics import accuracy_score, classification_report, confusion_
11 from textblob import TextBlob
12 from transformers import pipeline
13 import json
14
15 # Load dataset
```

# FASTER CODE GENERATION



JIE QI

Generate a Python script that analyzes airline reviews before and after COVID-19 and trains an SVM model to predict whether a passenger would recommend an airline.

↔ Data Augmentation Reviews

```
1 import pandas as pd
2 import numpy as np
3 import random
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 from sklearn.model_selection import train_test_split
7 from sklearn.preprocessing import LabelEncoder, StandardScaler
8 from sklearn.svm import SVC
9 from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
10 from textblob import TextBlob
11 from transformers import pipeline
12
13 # Load dataset
14 df = pd.read_csv("/mnt/data/Airline_review.csv")
15 df["Review Date"] = pd.to_datetime(df["Review Date"], errors="coerce")
```

# AUTOMATED MODEL EVALUATION



JIE QI

Write a Python script that automates the evaluation of an SVM model trained on airline review data.

↔ Data Augmentation Reviews

```
1 import pandas as pd
2 import numpy as np
3 import random
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 from sklearn.model_selection import train_test_split, GridSearchCV
7 from sklearn.preprocessing import LabelEncoder, StandardScaler
8 from sklearn.svm import SVC
9 from sklearn.metrics import accuracy_score, classification_report, confusion_
10 from textblob import TextBlob
11 from transformers import pipeline
12 import json
13
14 # Load dataset
15 df = pd.read_csv("/mnt/data/Airline_review.csv")
```

# DEMO: EFFICIENT DATA PREPROCESSING



SU TONG

Write me a code that do preprocessing tasks such as date conversion

Certainly! Below is a Python code snippet using the `pandas` library to handle several common data preprocessing tasks including date conversion, along with some additional features like normalizing text data and handling missing values more systematically:

```
python
import pandas as pd

# Example DataFrame with mixed data types and issues
data = {
    'Date': ['2023/01/01', '02-01-2023', '2023.01.03', None],
    'Temperature': [None, 22, 21, 20],
    'Comments': ['It was a good day.', 'rainy DAY', 'Sunny, clear skies!', None]
}

df = pd.DataFrame(data)

# Convert 'Date' to datetime, handling different formats
df['Date'] = pd.to_datetime(df['Date'], errors='coerce', dayfirst=True)

# Fill missing numeric values with the mean of the column
df['Temperature'].fillna(df['Temperature'].mean(), inplace=True)

# Normalize text in 'Comments': make all lowercase, remove trailing punctuation
df['Comments'] = df['Comments'].str.lower().str.replace(r'[^\w\s]', '', regex=True)

print(df)
```

This script performs the following preprocessing tasks:

# DEMO: HANDLING MISSING VALUES



CHONG GUI

In our dataset, the columns, "Inflight Entertainment", "Wifi & Connectivity" have missing values. Can code a statistical method to help us impute the missing values.

< 1/2 >

You can use **Multiple Imputation by Chained Equations (MICE)** or **K-Nearest Neighbors (KNN)** **imputation**, but a simpler and effective statistical approach is to use the **mean**, **median**, or **mode** base on the distribution of the data. Below is a Python script using **sklearn's SimpleImputer** to fill missing values based on different strategies.

```
python
import pandas as pd
from sklearn.impute import SimpleImputer

# Sample dataset
data = {
    "Inflight Entertainment": [5, 4, None, 3, 5, None, 4],
    "Wifi & Connectivity": [None, 3, 2, None, 4, 5, None]
}

df = pd.DataFrame(data)

# Choose an imputation strategy: 'mean', 'median', or 'most_frequent' (mode)
imputer = SimpleImputer(strategy="median") # Change to 'mean' or 'most_frequent' if needed

# Apply imputation to missing values
df[["Inflight Entertainment", "Wifi & Connectivity"]] = imputer.fit_transform(df[["Inflight Entertainment", "Wifi & Connectivity"]])

print(df)
```

# DEMO: HANDLING MISSING VALUES

You mentioned K-Nearest Neighbors imputation, can you generate a code for it as well and tell me the advantages of it over median imputation

Yes! Below is a Python script using **K-Nearest Neighbors (KNN) imputation**, which fills in missing values based on the nearest neighbors' values.

## KNN Imputation Code

```
python

import pandas as pd
import numpy as np
from sklearn.impute import KNNImputer

# Sample dataset with missing values
data = {
    "Inflight Entertainment": [5, 4, np.nan, 3, 5, np.nan, 4],
    "Wifi & Connectivity": [np.nan, 3, 2, np.nan, 4, 5, np.nan]
}

df = pd.DataFrame(data)

# KNN Imputer: 'n_neighbors' determines how many nearest neighbors to consider
knn_imputer = KNNImputer(n_neighbors=3) # Adjust 'n_neighbors' as needed

# Apply KNN imputation
df_imputed = pd.DataFrame(knn_imputer.fit_transform(df), columns=df.columns)

print(df_imputed)
```

## Advantages of KNN Imputation Over Median Imputation:

### 1. More Context-Aware

- Median imputation replaces all missing values with a single number, ignoring relationships between features.
- KNN considers patterns in the data, filling missing values with those from the most similar observations.

### 2. Preserves Data Distribution

- Median imputation may distort distributions by artificially concentrating values around the median.
- KNN maintains the natural variance and structure of the data.

### 3. Handles Complex Missing Patterns

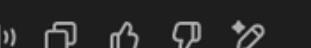
- If missing values depend on multiple factors, KNN can infer them better than a single-value imputation method.

### 4. More Accurate for Continuous Data

- Median works well for skewed distributions but loses important nuances in continuous variables.
- KNN can generate more precise estimates, especially when data points are closely related.

However, KNN is computationally expensive for large datasets, as it needs to compute distances between observations. If speed is a concern, median imputation may be a better choice.

Would you like to test this on your dataset? 😊



CHONG GUI

# DEMO: BALANCING THE DATASET

“Help write code to implement some form of data augmentation to produce synthetic data to ensure fair representation of data from both periods.”



JIE QI

I'll generate synthetic post-COVID reviews using **AI-based text augmentation** (GPT) and **basic text manipulation** (synonym replacement & sentence shuffling). The goal is to create additional reviews while maintaining a natural writing style.

Here's a Python script to do this:

```
</> Data Augmentation Reviews\n\n1 import pandas as pd\n2 import random\n3 from textblob import TextBlob\n4 from transformers import pipeline\n5\n6 # Load dataset\n7 df = pd.read_csv("/mnt/data/Airline_review.csv")\n8 df["Review Date"] = pd.to_datetime(df["Review Date"], errors="coerce")\n9\n10 # Define COVID cutoff\n11 date_cutoff = pd.to_datetime("2020-03-01")\n12 post_covid_df = df[df["Review Date"] >= date_cutoff]\n13\n14 # AI-based text generation\n15 text_generator = pipeline("text-generation", model="bert2")
```

# DEMO: TRAINING SUPPORT VECTOR MACHINE (SVM)

Based on our dataset, I want to predict airline recommendations before and after COVID-19. However, I only want to use a Support Vector Machine (SVM) model. Is it possible for you to provide me with the Python code for training the SVM model, along with its most ideal / optimised hyperparameters? Please do explain each portion of the code as well.

Yes! I can provide you with Python code to train an SVM model for predicting airline recommendations based on sentiment analysis before and after COVID-19. Below is a step-by-step breakdown of the process:

WEI TING



When it comes to training the support vector machine SVM model,

# DEMO: TRAINING SUPPORT VECTOR MACHINE (SVM)

```
# Set up a pipeline for TF-IDF feature extraction and SVM classification
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english', max_features=5000)), # Convert text to r
    ('svm', SVC(kernel='rbf', probability=True)) # Support Vector Machine classifier
])
```

## 3. Building a Pipeline

- Uses `TfidfVectorizer` to convert text into numerical format.
- Uses `SVC` (Support Vector Classifier) with an RBF kernel.

WEI TING



# DEMO: TRAINING SUPPORT VECTOR MACHINE (SVM)

```
# Define the hyperparameter grid for tuning
param_grid = {
    'svm_C': [0.1, 1, 10],    # Regularization parameter
    'svm_gamma': [0.01, 0.1, 1], # Kernel coefficient for 'rbf' kernel
    'svm_kernel': ['rbf']    # Radial basis function kernel
}

# Perform GridSearchCV to find the best parameters
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Get the best parameters
print(f"Best parameters: {grid_search.best_params_}")

# Train the SVM model with the best parameters
best_model = grid_search.best_estimator_
```

WEI TING



# DEMO: TRAINING SUPPORT VECTOR MACHINE (SVM)

## 4. Hyperparameter Tuning

- Performs a grid search to find the best `C` (regularization) and `gamma` (kernel coefficient).

WEI TING



# DEMO: EVALUATING SUPPORT VECTOR MACHINE (SVM)

please name me some of the best ways in which I can evaluate the performance of the SVM model

Evaluating the performance of your **SVM** model is crucial to ensure its effectiveness in predicting airline recommendations. Here are some of the best evaluation methods:

WEI TING



# DEMO: EVALUATING SUPPORT VECTOR MACHINE (SVM)

## 📌 Which Evaluation Metric to Use?

Scenario	Best Metrics
Balanced dataset	Accuracy, F1-score
Imbalanced dataset	F1-score, MCC, Cohen's Kappa
Model interpretation	Confusion Matrix, ROC-AUC
Generalization check	Cross-validation score

Would you like me to help visualize or further optimize your SVM model? 🚀

WEI TING



# DEMO: EVALUATING SUPPORT VECTOR MACHINE (SVM)

## 1 Accuracy

- **Definition:** Measures the percentage of correctly classified instances out of the total instances.
- **Formula:**

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Best Used When:** The dataset is **balanced** (equal distribution of classes).
- **Implementation in Python:**

python

 Copy  Edit

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
```

WEI TING



# DEMO: EVALUATING SUPPORT VECTOR MACHINE (SVM)

## 4 ROC Curve & AUC Score (For Probability-Based Evaluation)

- **ROC Curve (Receiver Operating Characteristic):** Plots **true positive rate (TPR)** vs. **false positive rate (FPR)**.
- **AUC Score (Area Under Curve):** Measures the probability that a randomly chosen positive instance ranks higher than a randomly chosen negative instance.
  - Closer to 1 → better model.
  - Closer to 0.5 → random guessing.
- **Implementation in Python:**

```
python
from sklearn.metrics import roc_curve, auc

y_scores = best_model.decision_function(X_test) # Get probability scores
fpr, tpr, _ = roc_curve(y_test, y_scores)
roc_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, color="blue", lw=2, label=f"ROC curve (AUC = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], color="gray", linestyle="--") # Random guessing line
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend(loc="lower right")
plt.show()
```

WEI TING



# PROS OF GAI



## Preprocess Data

Remove irrelevant or noisy data.

## Generate & Convert Code

Efficient generation and conversion of code

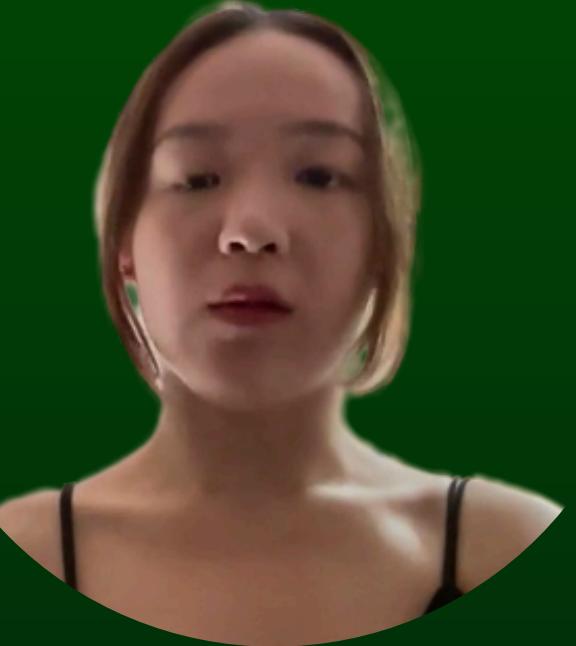
## Model Training & Evaluation

Training and evaluating the SVM model's performance effectively



**SHUN MEI**

# CONS OF GENAI



- 01** **Goes Against Data Protection Policies**
- 02** **Overreliance**
- 03** **Bias**

SHUN MEI