INSTALLING LIBRARIES

```
!pip install langdetect
!pip install deep-translator
!pip install emoji spacy
!pip install presidio-analyzer presidio-anonymizer spacy
!python -m spacy download en_core_web_lg
```

```
Requirement already satisfied: typer<1.0.0,>=0.3.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (0.15
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (4.67
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.1.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from spacy) (75.2.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (24.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.11/dist-packages (from cryptography<44.1->pr
Requirement already satisfied: language-data>=1.2 in /usr/local/lib/python3.11/dist-packages (from langcodes<4.0
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!
Requirement already satisfied: pydantic-core==2.33.1 in /usr/local/lib/python3.11/dist-packages (from pydantic!=
Requirement already satisfied: typing-extensions>=4.12.2 in /usr/local/lib/python3.11/dist-packages (from pydant
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydanti
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from request
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from thinc<8.4.0,>
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.11/dist-packages (from thinc<8
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from wease
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.11/dist-packages (from weasel<
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->spacy) (
Collecting requests-file>=1.4 (from tldextract->presidio-analyzer)
  Downloading requests_file-2.1.0-py2.py3-none-any.whl.metadata (1.7 kB)
Requirement already satisfied: filelock>=3.0.8 in /usr/local/lib/python3.11/dist-packages (from tldextract->pres
Requirement already satisfied: pycparser in /usr/local/lib/python3.11/dist-packages (from cffi>=1.12->cryptograp
Requirement already satisfied: marisa-trie>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from language-data
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.1
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packages (from smart-open<8.0.0,>=5.2.1->
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0
Downloading presidio_analyzer-2.2.358-py3-none-any.whl (114 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 114.9/114.9 kB 4.0 MB/s eta 0:00:00
Downloading presidio_anonymizer-2.2.358-py3-none-any.whl (31 kB)
Downloading phonenumbers-8.13.55-py2.py3-none-any.whl (2.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.6/2.6 MB 35.4 MB/s eta 0:00:00
Downloading tldextract-5.2.0-py3-none-any.whl (106 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 106.3/106.3 kB 7.0 MB/s eta 0:00:00
Downloading requests_file-2.1.0-py2.py3-none-any.whl (4.2 kB)
Installing collected packages: phonenumbers, requests-file, tldextract, presidio-anonymizer, presidio-analyzer
Successfully installed phonenumbers-8.13.55 presidio-analyzer-2.2.358 presidio-anonymizer-2.2.358 requests-file-
Collecting en-core-web-lg==3.8.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_lg-3.8.0/en_core_web_lg-3.
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 400.7/400.7 MB 4.0 MB/s eta 0:00:00
Installing collected packages: en-core-web-lg
Successfully installed en-core-web-lg-3.8.0
✓ Download and installation successful
You can now load the package via spacy.load('en_core_web_lg')
⚠ Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in
order to load all the package's dependencies. You can do this by selecting the
'Restart kernel' or 'Restart runtime' option.
```

Importing Libraries

```
import pandas as pd
from tqdm import tqdm
```

Data Preprocessing - Basic

```
# reading csv file
df=pd.read_csv("combined_emails_with_natural_pii.csv")
```

```
# displaying shape, columns, columns-dtypes
print(f"SHAPE OF THE DATA FRAME : {df.shape}")
print(f"COLUMNS PRESENT IN DATA FRAME : {df.columns}")
print("DATA TYPES FOR COLUMNS")
print(df.dtypes)
```

```
SHAPE OF THE DATA FRAME : (24000, 2)
COLUMNS PRESENT IN DATA FRAME : Index(['email', 'type'], dtype='object')
DATA TYPES FOR COLUMNS
email    object
type     object
dtype: object
```

```
# detecting the presence of missing values, duplicate rows
print(f"TOTAL NUMBER OF MISSING VALUES : {df.isna().sum().sum()}")
print("COLUMN WISE NUMBER OF MISSING VALUES")
print(df.isna().sum())
print(f"NUMBER OF DUPLICATE ROWS PRESENT IN DATA FRAME : {df[df.duplicated()].shape}")
print("ACTION ON DROPPING DUPLICATE VALUES")
if df[df.duplicated()].shape[0]>0:
  df=df.drop_duplicates()
  print("ACTION(DROPPED DUPLICATES)")
else:
  print("NO ACTION IS PERFORMED SINCE NO DUPLICATE ROWS FOUND")
```

```
TOTAL NUMBER OF MISSING VALUES : 0
COLUMN WISE NUMBER OF MISSING VALUES
email    0
type     0
dtype: int64
NUMBER OF DUPLICATE ROWS PRESENT IN DATA FRAME : (0, 2)
ACTION ON DROPPING DUPLICATE VALUES
NO ACTION IS PERFORMED SINCE NO DUPLICATE ROWS FOUND
```

```
# Number of unique values and unique values present in the column 'type'
print(f"NUMBER OF UNIQUE VALUES IN COLUMN 'type' :{df['type'].nunique()}")
print(f"UNIQUE VALUES PRESENT IN COLUMN 'type' :{list(df['type'].unique())}")
```

```
NUMBER OF UNIQUE VALUES IN COLUMN 'type' :4
UNIQUE VALUES PRESENT IN COLUMN 'type' :['Incident', 'Request', 'Problem', 'Change']
```

```
# Checking whether the data set is balanced or not
print("FREQUENCIES FOR EACH UNIQUE VALUE IN COLUMN 'type'")
df['type'].value_counts()
```

```
FREQUENCIES FOR EACH UNIQUE VALUE IN COLUMN 'type'
```

type	count
Incident	9586
Request	6860
Problem	5037
Change	2517

dtype: int64

```
print("THIS IS AN IMBALANCED DATA SET")
```

```
THIS IS AN IMBALANCED DATA SET
```

```
# printing a single value from column 'email'
df['email'][0]
```

```
'Subject: Unvorhergesehener Absturz der Datenanalyse-Plattform\n\nDie Datenanalyse-Plattform brach unerwartet ab, da die Speicheroberfläche zu gering war My name is Sophia Rossi.. Ich habe versucht, Laravel 8 und meinen MacBook Pro neu zu starten, aber das Problem behält sich bei. Ich benötige Ihre Unterstützung, um diesen Fehler zu beheben. You can reach me at janesmith@company.com.'
```

```
from langdetect import detect

def detect_languages_offline(df):
    language_list = []
    for i in tqdm(df['email'], desc="Detecting emails Language"):
        try:
            language_list.append(detect(i))
        except:
            language_list.append("unknown")

    frequencyDict = {}
    for lang in set(language_list):
        frequencyDict[lang] = language_list.count(lang)

    return frequencyDict


frequencyDict=detect_languages_offline(df)
print(frequencyDict)
```

```
cting emails Language: 100%|██████████| 24000/24000 [01:58<00:00, 202.75it/s]{'es': 812, 'pt': 474, 'nl': 5, 'it':
```

```
import pandas as pd
from langdetect import detect
from deep_translator import GoogleTranslator
from tqdm import tqdm

# Translate all non-English emails to English
def translate_languages(df):
    translated_emails = []

    for text in tqdm(df['email'], desc="Translating emails"):
        try:
            detected_lang = detect(text)
            if detected_lang != 'en':
                translated_text = GoogleTranslator(source=detected_lang, target='en').translate(text)
```

```
            else:
                translated_text = text
        except Exception as e:
            # Fallback in case of error
            translated_text = text
        translated_emails.append(translated_text)

    df['email'] = translated_emails
    return df
```

```
print("BEFORE TRANSLATION : ")
lang_count=detect_languages_offline(df)
print(lang_count)
df=translate_languages(df)
lang_count=detect_languages_offline(df)
print("AFTER TRANSLATION : ")
print(lang_count)
```

```
BEFORE TRANSLATION :
Detecting emails Language: 100%|██████████| 24000/24000 [01:53<00:00, 210.87it/s]
{'es': 812, 'pt': 474, 'nl': 4, 'de': 6926, 'fr': 485, 'it': 1, 'en': 15298}
Translating emails: 100%|██████████| 24000/24000 [2:31:37<00:00,  2.64it/s]
Detecting emails Language: 100%|██████████| 24000/24000 [01:49<00:00, 219.38it/s]AFTER TRANSLATION :
{'en': 23991, 'nl': 2, 'de': 6, 'fr': 1}
```

```
df.to_csv("modified_csv.csv")
```

```
df['email'][0]
```

```
'Subject: unforeseen crash of the data analysis platform\n\nThe data analysis platform broke off unexpectedly bec
ause the memory surface was too small My name is sophia rossi .. I tried to start Laravel 8 and my MacBook Pro ag
ain, but the problem retains. I need your support to fix this mistake. You Can Reach Me at janesmith@company.co
m '
```

```
print("BEFORE TRANSLATION : ")
lang_count=detect_languages_offline(df)
print(lang_count)
df=translate_languages(df)
lang_count=detect_languages_offline(df)
print("AFTER TRANSLATION : ")
print(lang_count)
```

```
BEFORE TRANSLATION :
Detecting emails Language: 100%|██████████| 24000/24000 [01:47<00:00, 224.13it/s]
{'en': 23989, 'nl': 3, 'de': 6, 'fr': 2}
Translating emails: 100%|██████████| 24000/24000 [01:57<00:00, 205.10it/s]
Detecting emails Language: 100%|██████████| 24000/24000 [01:47<00:00, 222.41it/s]AFTER TRANSLATION :
{'en': 23999, 'nl': 1}
```

```
df.to_csv("modified(1).csv")
```

```
print("BEFORE TRANSLATION : ")
lang_count=detect_languages_offline(df)
print(lang_count)
df=translate_languages(df)
lang_count=detect_languages_offline(df)
print("AFTER TRANSLATION : ")
print(lang_count)
```

```
BEFORE TRANSLATION :
Detecting emails Language: 100%|██████████| 24000/24000 [01:21<00:00, 295.39it/s]
{'en': 23991, 'nl': 1, 'de': 5, 'fr': 1, 'es': 2}
```

```
Translating emails: 100%|██████████| 24000/24000 [01:24<00:00, 283.44it/s]
Detecting emails Language: 100%|██████████| 24000/24000 [01:22<00:00, 291.35it/s]AFTER TRANSLATION :
{'fr': 1, 'en': 23999}
```

```
df.to_csv("modified(2).csv")
```

```
df=pd.read_csv("modified(2).csv")
```

ADVANCED DATA PREPROCESSING

```
# viewing sample data set with 5 rows
df.sample(5)
```

	Unnamed: 0	email	type
4380	4380	Subject: Request for Software Assistance\n\nAn...	Incident
7856	7856	Subject: Unauthorized Access to Medical Data D...	Incident
21763	21763	Subject: Urgent: Jira Ticket System Disruption...	Incident
16979	16979	Subject: Request for Details on Salesforce CRM...	Request
6956	6956	Subject: Data Analytics Dashboard Sudden Crash...	Problem

```
# dropping the column Unnamed: 0
df=df.drop(['Unnamed: 0'],axis=1)
```

```
# viewing columns
df.columns
```

```
Index(['email', 'type'], dtype='object')
```

```
# converting email to lower case
df['email']=df['email'].str.lower()
print("SAMPLE DOCUMENT AFTER LOWER CASING")
df.head(1)
```

```
SAMPLE DOCUMENT AFTER LOWER CASING
```

	email	type
0	subject: unforeseen crash of the data analysis...	Incident

```
# Removing html tags, url's, punctuation's and encoding the emoji's if used
```

```
import re
import string
import emoji
import pandas as pd

def clean_email_text(df):
    cleaned_emails = []

    for text in df['masked_email']:
        # 1. Remove HTML tags
        text = re.sub(r'<.*?>', '', text)

        # 2. Remove escape characters (like \n, \t, \r)
        text = re.sub(r'[\n\r\t]', ' ', text)
```

```
        # 3. Remove URLs
        text = re.sub(r'http\S+|www\.\S+', '', text)

        # 4. Remove punctuation
        text = ''.join(ch for ch in text if ch not in string.punctuation)

        # 5. Encode emojis (convert emojis to text like ':smile:')
        text = emoji.demojize(text)

        # Optional: Remove multiple spaces
        text = re.sub(r'\s+', ' ', text).strip()

        cleaned_emails.append(text)

    df['email'] = cleaned_emails
    return df


df=clean_email_text(df)


df['email'][0]
```

```
'subject: unforeseen crash of the data analysis platform the data analysis platform broke off unexpectedly becaus
e the memory surface was too small my name is sophia rossi .. i tried to start laravel 8 and my macbook pro agai
n, but the problem retains. i need your support to fix this mistake. you can reach me at janesmith@company.co
m '
```

```
from presidio_analyzer import AnalyzerEngine, RecognizerRegistry, PatternRecognizer, Pattern
from presidio_anonymizer import AnonymizerEngine
from tqdm import tqdm
import pandas as pd

# Create analyzer
analyzer = AnalyzerEngine()

# Add custom recognizers (for Aadhar, CVV, Expiry)
aadhar_pattern = Pattern(name="Aadhar Pattern", regex=r"\b\d{4}[\s\-]?\d{4}[\s\-]?\d{4}\b", score=0.85)
cvv_pattern = Pattern(name="CVV Pattern", regex=r"\b\d{3}\b", score=0.8)
expiry_pattern = Pattern(name="Expiry Pattern", regex=r"\b(0[1-9]|1[0-2])/?(\d{2}|\d{4})\b", score=0.8)

aadhar_recognizer = PatternRecognizer(supported_entity="AADHAR_NUM", patterns=[aadhar_pattern])
cvv_recognizer = PatternRecognizer(supported_entity="CVV_NO", patterns=[cvv_pattern])
expiry_recognizer = PatternRecognizer(supported_entity="EXPIRY_NO", patterns=[expiry_pattern])

# Register them
analyzer.registry.add_recognizer(aadhar_recognizer)
analyzer.registry.add_recognizer(cvv_recognizer)
analyzer.registry.add_recognizer(expiry_recognizer)

# Map Presidio entity names to required ones
entity_label_map = {
    "PERSON": "full_name",
    "EMAIL_ADDRESS": "email",
    "PHONE_NUMBER": "phone_number",
    "DATE_TIME": "dob",
    "CREDIT_CARD": "credit_debit_no",
    "AADHAR_NUM": "aadhar_num",
    "CVV_NO": "cvv_no",
    "EXPIRY_NO": "expiry_no"
}

# Function to mask a single email
def mask_email_content(text):
    results = analyzer.analyze(text=text, entities=[], language='en')
    results = sorted(results, key=lambda r: r.start, reverse=True)
```

```
    for r in results:
        entity_name = entity_label_map.get(r.entity_type, None)
        if entity_name:
            text = text[:r.start] + f"[{entity_name}]" + text[r.end:]

    return text
```

```
WARNING:presidio-analyzer:Recognizer not added to registry because language is not supported by registry - CreditC
WARNING:presidio-analyzer:Recognizer not added to registry because language is not supported by registry - CreditC
WARNING:presidio-analyzer:Recognizer not added to registry because language is not supported by registry - CreditC
WARNING:presidio-analyzer:Recognizer not added to registry because language is not supported by registry - EsNifRe
WARNING:presidio-analyzer:Recognizer not added to registry because language is not supported by registry - EsNieRe
WARNING:presidio-analyzer:Recognizer not added to registry because language is not supported by registry - ItDrive
WARNING:presidio-analyzer:Recognizer not added to registry because language is not supported by registry - ItFisca
WARNING:presidio-analyzer:Recognizer not added to registry because language is not supported by registry - ItVatCo
WARNING:presidio-analyzer:Recognizer not added to registry because language is not supported by registry - ItIdent
WARNING:presidio-analyzer:Recognizer not added to registry because language is not supported by registry - ItPassp
WARNING:presidio-analyzer:Recognizer not added to registry because language is not supported by registry - PlPesel
```

```
def mask_dataframe_emails(df, text_column='email'):
    tqdm.pandas(desc="Masking PII/PCI with Presidio")
    df['masked_email'] = df[text_column].progress_apply(mask_email_content)
    return df
```

```
df = mask_dataframe_emails(df)
```

```
Masking PII/PCI with Presidio: 100%|██████████| 24000/24000 [11:49<00:00, 33.85it/s]
```

```
df['masked_email'][2]
```

```
'subject: data analytics for investment i am contacting you to request information on data analytics tools that c
an be utilized with the eclipse ide for enhancing investment optimization. i am seeking suggestions for tools tha
t can aid in making data-driven decisions. particularly, i am interested in tools that can manage large datasets
and offer advanced analytics features. these tools should be compatible with the eclipse ide and can smoothly int
egrate into my workflow you can reach me at [email].. key features i am interested in include data visualization,
predictive modeling, and machine learning capabilities. i would greatly appreciate any recommendations or advice
on how to begin with data analytics for investment optimization using the eclipse ide. my name is [full_name].'
```

```
df=clean_email_text(df)
```

```
df['masked_email'][2]
```

```
'subject: data analytics for investment i am contacting you to request information on data analytics tools that c
an be utilized with the eclipse ide for enhancing investment optimization. i am seeking suggestions for tools tha
t can aid in making data-driven decisions. particularly, i am interested in tools that can manage large datasets
and offer advanced analytics features. these tools should be compatible with the eclipse ide and can smoothly int
egrate into my workflow you can reach me at [email].. key features i am interested in include data visualization,
predictive modeling, and machine learning capabilities. i would greatly appreciate any recommendations or advice
on how to begin with data analytics for investment optimization using the eclipse ide. my name is [full_name].'
```

```
df.columns
```

```
Index(['type', 'masked_email'], dtype='object')
```

```
df.to_csv("email.csv")
```

```
df=pd.read_csv("email.csv")
df.columns
```

```
Index(['Unnamed: 0', 'type', 'masked_email'], dtype='object')
```

```
df=df.drop(['Unnamed: 0'],axis=1)
```

Double-click (or enter) to edit

```
df=df.rename({"masked_email":'email'},axis=1)
```

```
df.columns
```

```
Index(['type', 'email'], dtype='object')
```

SOME NLP TECHNIQUES

```
import spacy
import pandas as pd
from tqdm import tqdm

# Load spaCy English model
nlp = spacy.load("en_core_web_sm")

def process_emails(df):
    lemmatized_nostop = []

    for email in tqdm(df['email'], desc="Lemmatizing without stopwords"):
        doc = nlp(email)

        # Lemmatize and remove stopwords + punctuation
        lemmas = [token.lemma_ for token in doc if not token.is_stop and not token.is_punct]
        lemmatized_nostop.append(' '.join(lemmas))

    # Add new column to DataFrame
    df['email_lemmatized_nostopwords'] = lemmatized_nostop

    return df
```

```
df=process_emails(df)
```

```
Lemmatizing without stopwords: 100%|██████████| 24000/24000 [09:10<00:00, 43.57it/s]
```

```
df=pd.read_csv("email_lemmatized_nostopwords.csv")
```

```
df=df.drop(['Unnamed: 0', 'email'],axis=1)
```

```
df['email_lemmatized_nostopwords'][2]
```

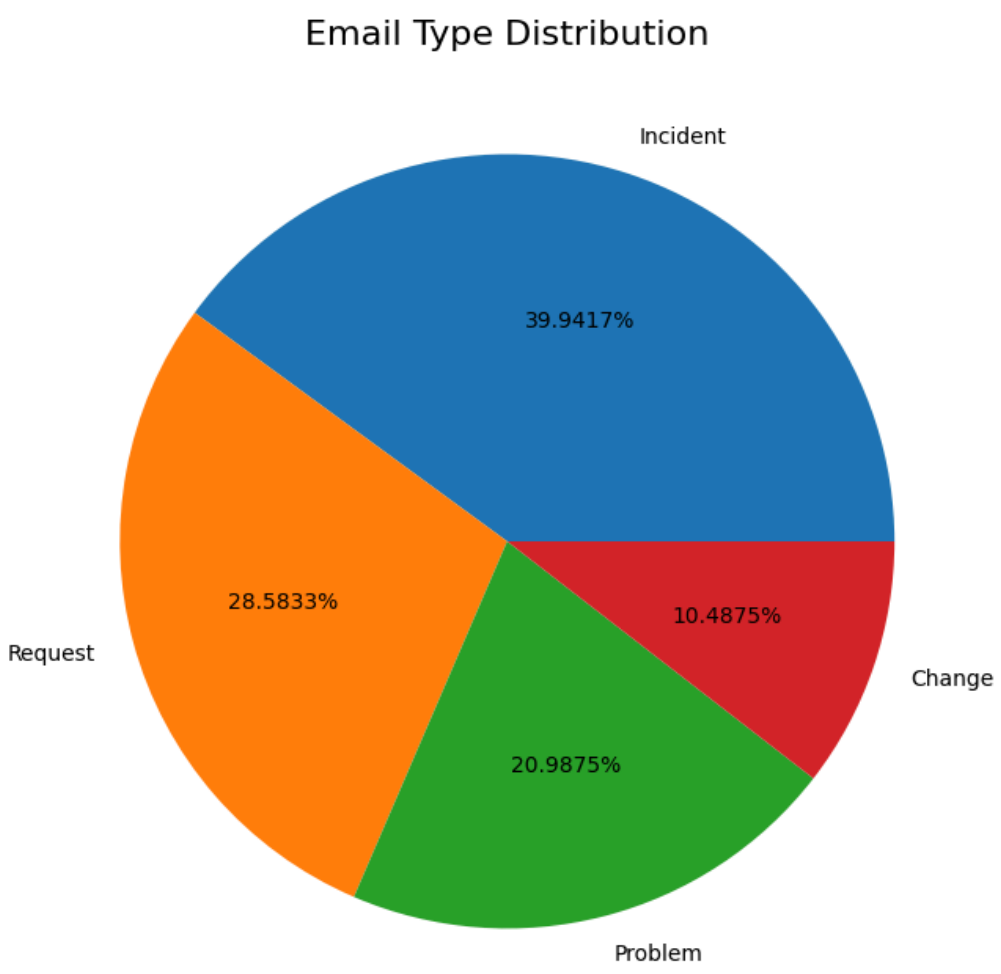
```
'subject data analytic investment contact request information datum analytic tool utilize eclipse ide enhance inv
estment optimization seek suggestion tool aid make data drive decision particularly interested tool manage large
dataset offer advanced analytic feature tool compatible eclipse ide smoothly integrate workflow reach email key f
eature interested include datum visualization predictive modeling machine learning capability greatly appreciate
recommendation advice begin data analytic investment optimization eclipse ide full_name'
```

```
df=df.rename({'email_lemmatized_nostopwords':'email'},axis=1)
```

Exploratory Data Analysis

```
def format_pct(pct):
    return f"{pct:.4f}%"
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(8,8))
plt.pie(df['type'].value_counts(),labels=df['type'].unique(),autopct=format_pct)
plt.title("Email Type Distribution", fontsize=16)
plt.show()
```




```
df=df.rename({'email_lemmatized_nostopwords':'email'},axis=1)
df.columns
```

```
Index(['type', 'email'], dtype='object')
```

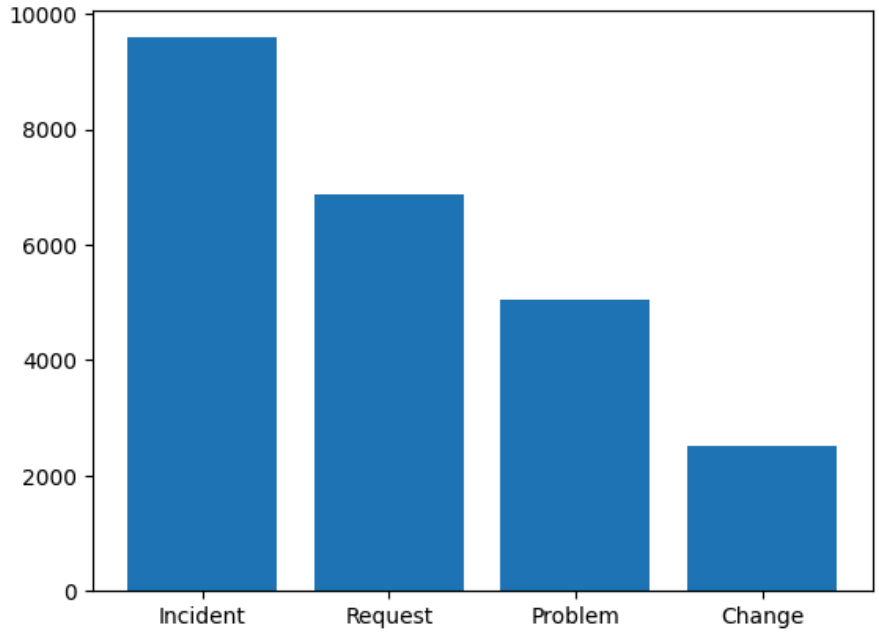
```
!pip install wordcloud
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.11/dist-packages (1.9.4)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.11/dist-packages (from wordcloud) (2.0.2)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from wordcloud) (11.1.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from wordcloud) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordc
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->word
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->word
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcl
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordc
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib->w
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->mat
```

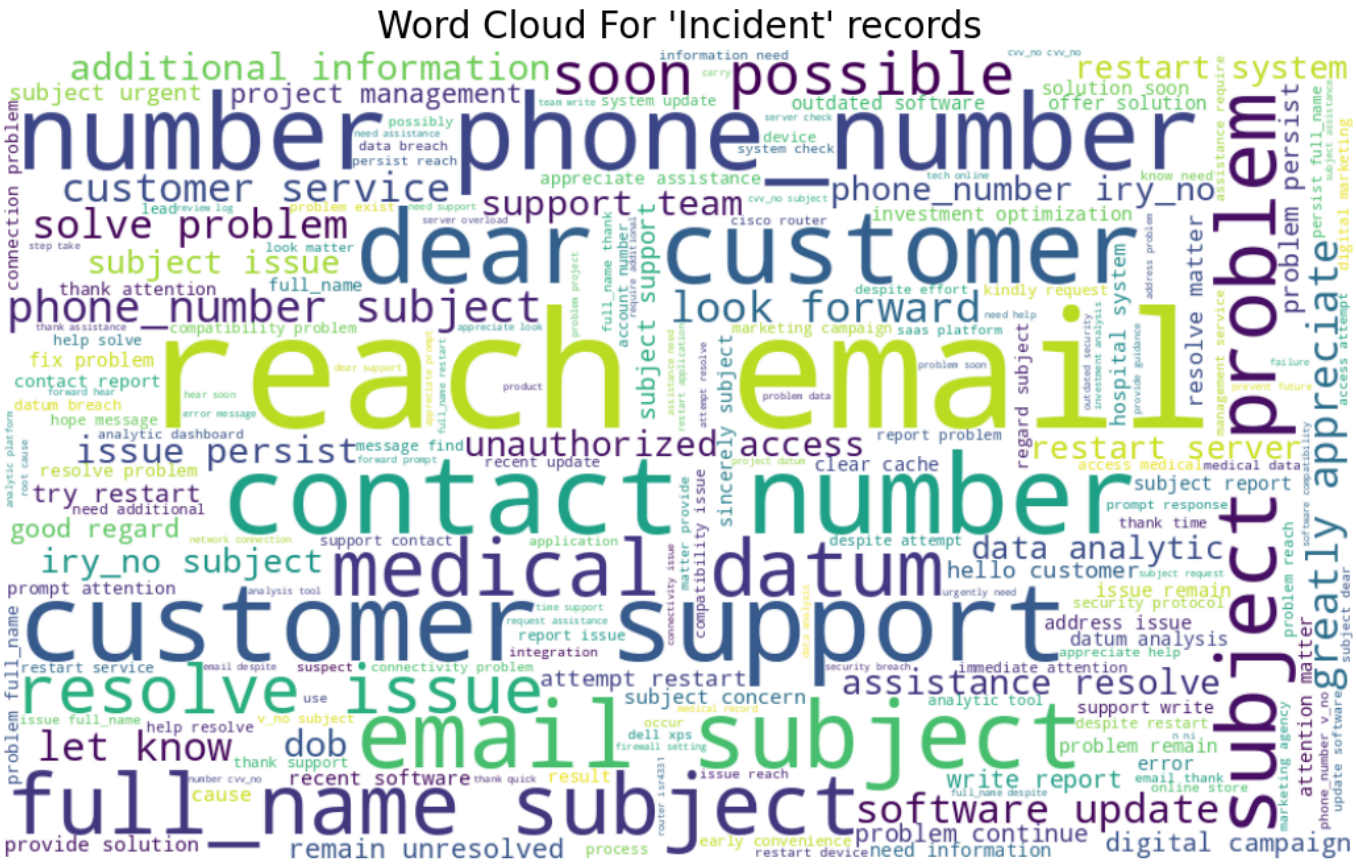
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

```
def plot_wordcloud_from_column(df, column='email', title='Word Cloud'):
    # Combine all text in the column
    text = ' '.join(df[column].dropna().astype(str))

    # Generate word cloud
    wordcloud = WordCloud(width=1000, height=600, background_color='white', colormap='viridis').generate(text)

    # Plot
    plt.figure(figsize=(15, 8))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(title, fontsize=20)
    plt.show()
```

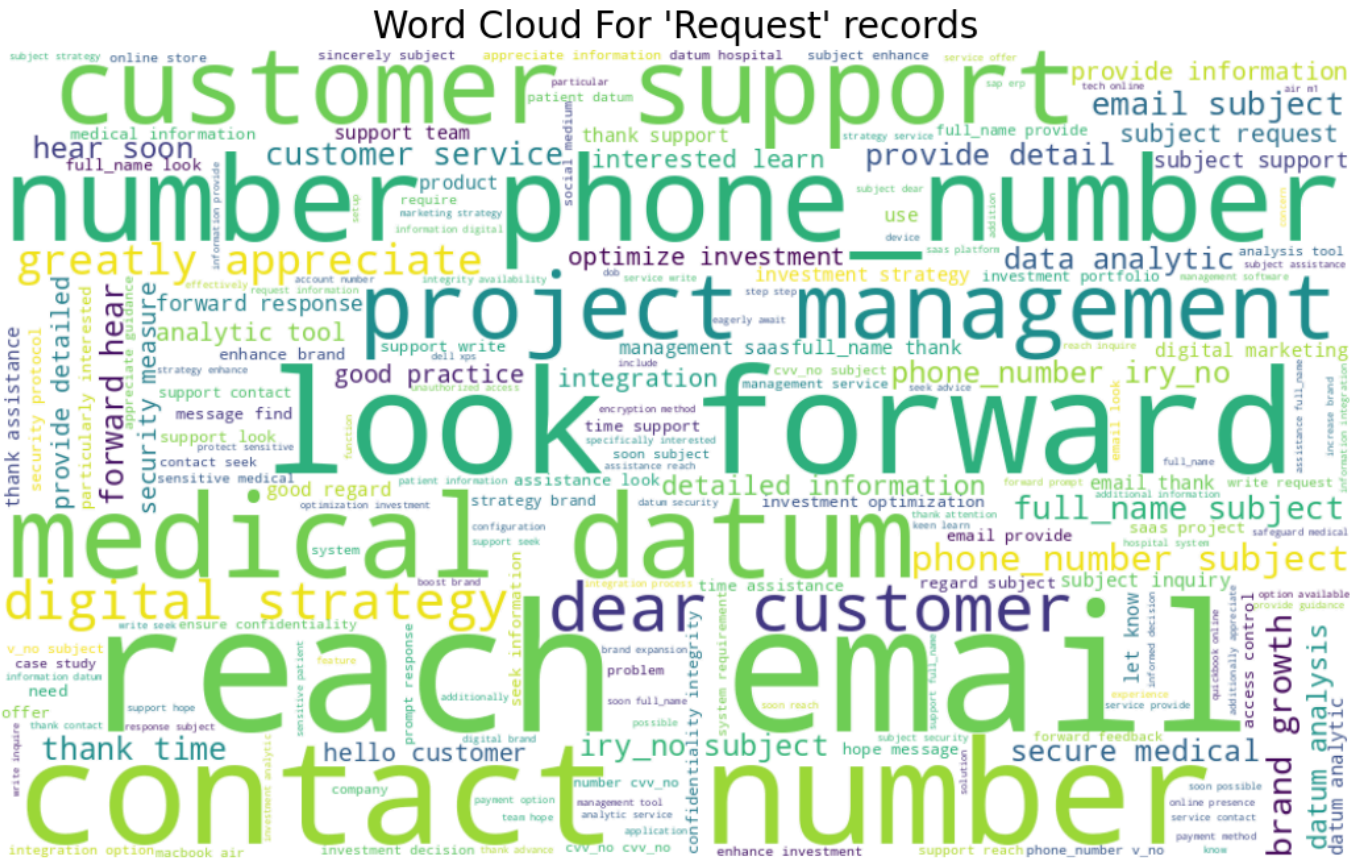
```
value_counts=df['type'].value_counts()
labels=value_counts.index
plt.bar(labels,value_counts)
plt.show()
```



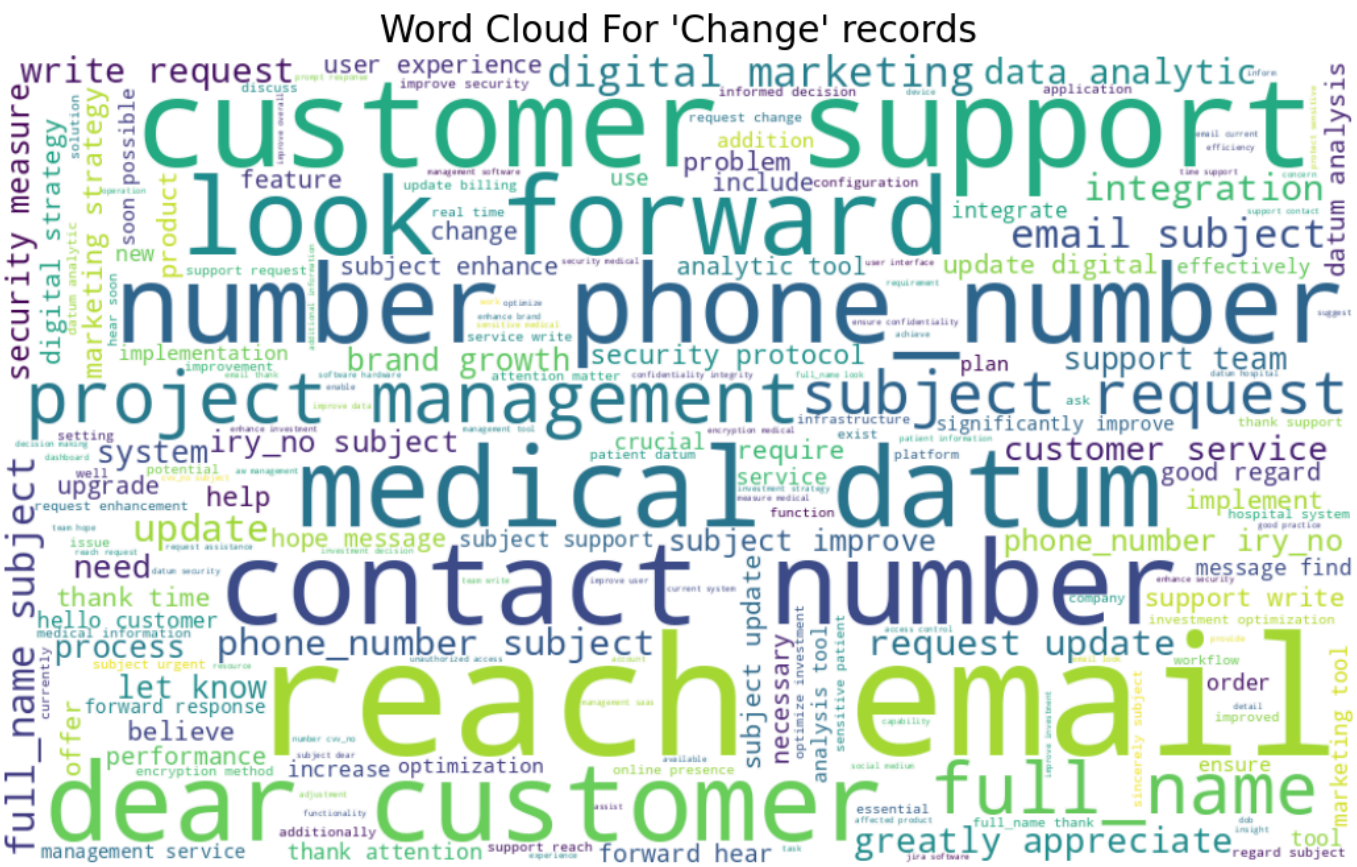

```
# Example usage
plot_wordcloud_from_column(df[df['type']=='Incident'],title="Word Cloud For 'Incident' records")
```



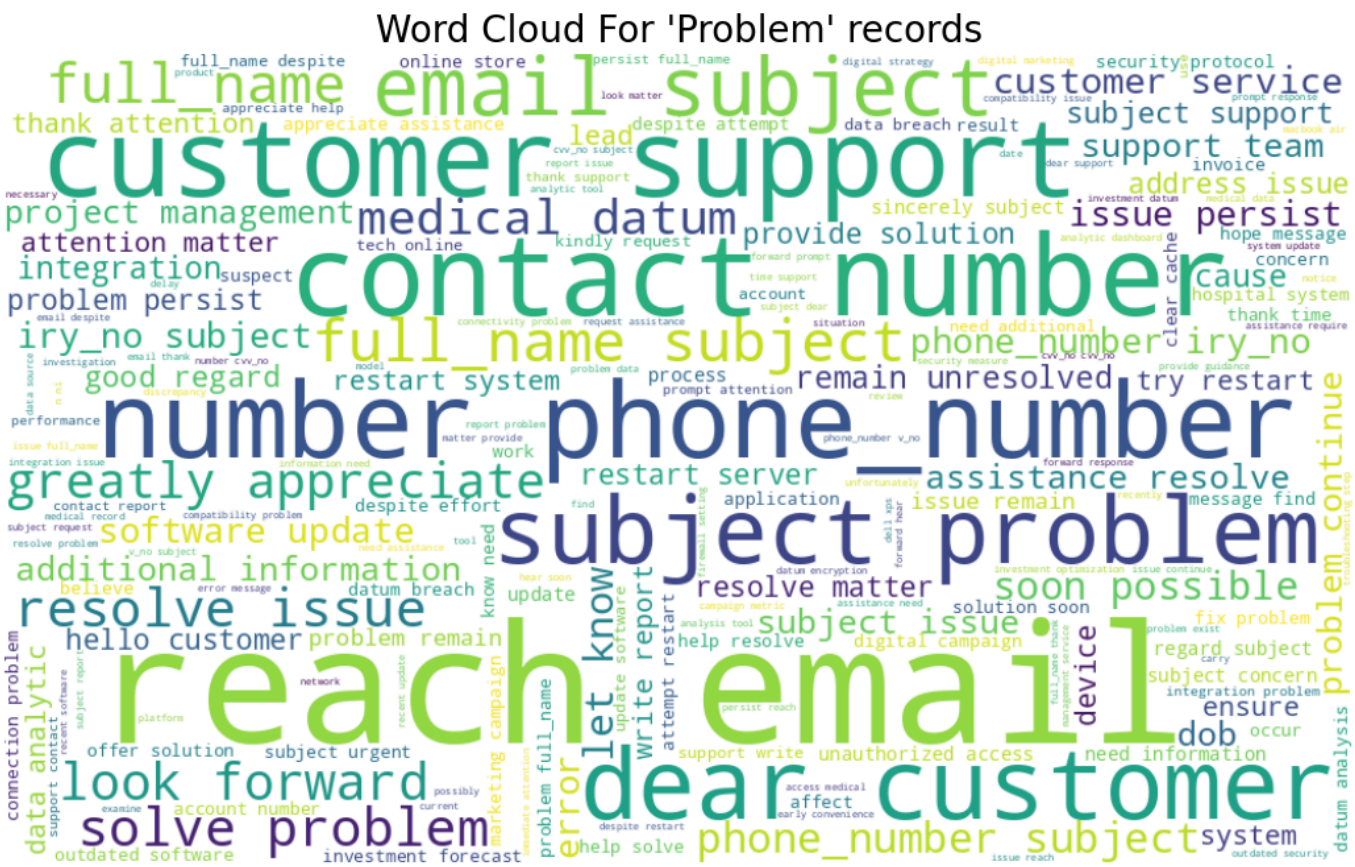
```
# Example usage
plot_wordcloud_from_column(df[df['type']=='Request'],title="Word Cloud For 'Request' records")
```




```
# Example usage
plot_wordcloud_from_column(df[df['type']=='Change'],title="Word Cloud For 'Change' records")
```




```
# Example usage
plot_wordcloud_from_column(df[df['type']=='Problem'],title="Word Cloud For 'Problem' records")
```



FEATURE ENGINEERING + MODEL SELECTION + MODEL IMPLEMENTATION + EVALUATION

```
df.columns
```

```
Index(['type', 'email'], dtype='object')
```

WITH TFIDF VECTORIZER

```
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import OrdinalEncoder
from sklearn.feature_extraction.text import *
from sklearn.metrics import classification_report, accuracy_score
import pandas as pd

# Prepare data
X = df['email']
y = df[['type']]

# Encode target
encoder = OrdinalEncoder()
y_encoded = encoder.fit_transform(y)

# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y_encoded.ravel(), test_size=0.2, random_state=42, stratify=y_encoded
)
```

```
y_encoded[2]
```

```
array([3.])
```

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer=TfidfVectorizer()
X_train_transformed=vectorizer.fit_transform(X_train)
x_test_transformed=vectorizer.transform(X_test)
model=MultinomialNB()
model.fit(X_train_transformed,y_train)
y_pred = model.predict(x_test_transformed)

print("MultinomialNB")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))
```

```
MultinomialNB
Accuracy: 0.7072916666666667
              precision    recall  f1-score   support

      Change       0.95      0.36      0.52       504
    Incident       0.64      0.97      0.77      1917
     Problem       0.68      0.10      0.18      1007
     Request       0.81      0.92      0.86      1372

    accuracy                           0.71      4800
   macro avg       0.77      0.59      0.58      4800
weighted avg       0.73      0.71      0.64      4800
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer=TfidfVectorizer()
X_train_transformed=vectorizer.fit_transform(X_train)
x_test_transformed=vectorizer.transform(X_test)
model=LogisticRegression(max_iter=1000)
model.fit(X_train_transformed,y_train)
y_pred = model.predict(x_test_transformed)

print("MultinomialNB")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))
```

```
MultinomialNB
Accuracy: 0.7702083333333334
              precision    recall  f1-score   support

      Change       0.93      0.81      0.87       504
    Incident       0.70      0.86      0.77      1917
     Problem       0.59      0.35      0.44      1007
     Request       0.91      0.94      0.92      1372

    accuracy                           0.77      4800
   macro avg       0.78      0.74      0.75      4800
weighted avg       0.76      0.77      0.76      4800
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Assuming X_train, X_test, y_train, y_test, encoder, df are already defined
```

```
# Vectorize the training and test data
vectorizer = TfidfVectorizer()
X_train_transformed = vectorizer.fit_transform(X_train)
X_test_transformed = vectorizer.transform(X_test)

# Train the LinearSVC model
model = LinearSVC()
model.fit(X_train_transformed, y_train)

# Predict on test data
y_pred = model.predict(X_test_transformed)

# Evaluation
print("LinearSVC")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))

# Predict on a new example from the DataFrame
print("PREDICTIONS USING LINEAR SVC")
print(df.head(3))

data = df['email'][100]
print(data)

# Fix: Pass a list of the string directly
need_prediction = vectorizer.transform([data])
prediction = model.predict(need_prediction)
print("Prediction for new email:", prediction[0])
```

```
LinearSVC
Accuracy: 0.77625
              precision    recall  f1-score   support

      Change       0.91      0.86      0.89       504
    Incident       0.71      0.84      0.77      1917
     Problem       0.59      0.39      0.47      1007
     Request       0.92      0.94      0.93      1372

    accuracy                           0.78      4800
   macro avg       0.78      0.76      0.76      4800
weighted avg       0.77      0.78      0.76      4800

PREDICTIONS USING LINEAR SVC
       type                                              email
0  Incident  subject unforeseen crash data analysis platfor...
1   Request  subject customer support inquiry seek informat...
2   Request  subject data analytic investment contact reque...
subject different campaign metric dear customer support sign report diversity campaign metric reach email error oc
Prediction for new email: 2.0
```

```
y_encoded[100]
```

```
array([2.])
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Assuming X_train, X_test, y_train, y_test, encoder, df are already defined

# Vectorize the training and test data
vectorizer = TfidfVectorizer()
X_train_transformed = vectorizer.fit_transform(X_train)
X_test_transformed = vectorizer.transform(X_test)
```

```
X_test_transformed = vectorizer.transform(X_test)

# Train the RandomForestClassifier model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_transformed, y_train)

# Predict on test data
y_pred = model.predict(X_test_transformed)

# Evaluation
print("RandomForestClassifier")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))

# Predict on a new example from the DataFrame
print("PREDICTIONS USING RANDOM FOREST")
print(df.head(3))

data = df['email'][10]
print(data)

# Predict
need_prediction = vectorizer.transform([data])
prediction = model.predict(need_prediction)
print("Prediction for new email:", prediction[0])

if prediction[0]==y_encoded[10]:
  print("=======================================================")
  print("CORRECT CLASSIFICATION")
  print("=======================================================")
else:
  print("=======================================================")
  print("WRONG CLASSIFICATION")
  print("=======================================================")
```

```
RandomForestClassifier
Accuracy: 0.78125
              precision    recall  f1-score   support

      Change       0.98      0.69      0.81       504
    Incident       0.68      0.98      0.81      1917
     Problem       0.94      0.22      0.35      1007
     Request       0.89      0.95      0.92      1372

    accuracy                           0.78      4800
   macro avg       0.87      0.71      0.72      4800
weighted avg       0.83      0.78      0.74      4800

PREDICTIONS USING RANDOM FOREST
       type                                              email
0  Incident  subject unforeseen crash data analysis platfor...
1   Request  subject customer support inquiry seek informat...
2   Request  subject data analytic investment contact reque...
subject issue data analytic platform insufficient ram allocation reach report issue data analytic platform platfor
Prediction for new email: 2.0
=======================================================
CORRECT CLASSIFICATION
=======================================================
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Assuming X_train, X_test, y_train, y_test, encoder, df, y_encoded are already defined
```

```
# Vectorize the training and test data
vectorizer = TfidfVectorizer()
X_train_transformed = vectorizer.fit_transform(X_train)
X_test_transformed = vectorizer.transform(X_test)

# Train the GradientBoostingClassifier model
model = GradientBoostingClassifier()
model.fit(X_train_transformed, y_train)

# Predict on test data
y_pred = model.predict(X_test_transformed)

# Evaluation
print("GradientBoostingClassifier")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))

# Predict on a new example from the DataFrame
print("PREDICTIONS USING GRADIENT BOOSTING")
print(df.head(3))

data = df['email'][0]
print(data)

# Predict
need_prediction = vectorizer.transform([data])
prediction = model.predict(need_prediction)
print("Prediction for new email:", prediction[0])

# Check if prediction is correct
if prediction[0] == y_encoded[0]:
    print("=====================================================")
    print("CORRECT CLASSIFICATION")
    print("=====================================================")
else:
    print("=====================================================")
    print("WRONG CLASSIFICATION")
    print("=====================================================")
```

```
GradientBoostingClassifier
Accuracy: 0.740625
              precision    recall  f1-score   support

      Change       0.90      0.73      0.81       504
    Incident       0.66      0.91      0.77      1917
     Problem       0.61      0.17      0.26      1007
     Request       0.86      0.93      0.89      1372

    accuracy                           0.74      4800
   macro avg       0.76      0.68      0.68      4800
weighted avg       0.73      0.74      0.70      4800

PREDICTIONS USING GRADIENT BOOSTING
       type                                              email
0  Incident  subject unforeseen crash data analysis platfor...
1   Request  subject customer support inquiry seek informat...
2   Request  subject data analytic investment contact reque...
subject unforeseen crash data analysis platform data analysis platform break unexpectedly memory surface small ful
Prediction for new email: 1.0
=====================================================
CORRECT CLASSIFICATION
=====================================================
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
```

```
import pandas as pd

# Assuming X_train, X_test, y_train, y_test, encoder, df, y_encoded are already defined

# Vectorize the training and test data
vectorizer = TfidfVectorizer()
X_train_transformed = vectorizer.fit_transform(X_train)
X_test_transformed = vectorizer.transform(X_test)

# Train the DecisionTreeClassifier model
model = DecisionTreeClassifier()
model.fit(X_train_transformed, y_train)

# Predict on test data
y_pred = model.predict(X_test_transformed)

# Evaluation
print("DecisionTreeClassifier")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))

# Predict on a new example from the DataFrame
print("PREDICTIONS USING DECISION TREE")
print(df.head(3))

data = df['email'][10]
print(data)

# Predict
need_prediction = vectorizer.transform([data])
prediction = model.predict(need_prediction)
print("Prediction for new email:", prediction[0])

# Check if prediction is correct
if prediction[0] == y_encoded[10]:
    print("=====================================================")
    print("CORRECT CLASSIFICATION")
    print("=====================================================")
else:
    print("=====================================================")
    print("WRONG CLASSIFICATION")
    print("=====================================================")
```

```
DecisionTreeClassifier
Accuracy: 0.693125
              precision    recall  f1-score   support

      Change       0.71      0.69      0.70       504
    Incident       0.69      0.72      0.70      1917
     Problem       0.45      0.41      0.43      1007
     Request       0.86      0.86      0.86      1372

    accuracy                           0.69      4800
   macro avg       0.68      0.67      0.67      4800
weighted avg       0.69      0.69      0.69      4800

PREDICTIONS USING DECISION TREE
       type                                              email
0  Incident  subject unforeseen crash data analysis platfor...
1   Request  subject customer support inquiry seek informat...
2   Request  subject data analytic investment contact reque...
subject issue data analytic platform insufficient ram allocation reach report issue data analytic platform platfor
Prediction for new email: 2.0
=====================================================
CORRECT CLASSIFICATION
=====================================================
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Assuming X_train, X_test, y_train, y_test, encoder, df, y_encoded are already defined

# Vectorize the training and test data
vectorizer = TfidfVectorizer()
X_train_transformed = vectorizer.fit_transform(X_train)
X_test_transformed = vectorizer.transform(X_test)

# Train the SGDClassifier model
model = SGDClassifier()
model.fit(X_train_transformed, y_train)

# Predict on test data
y_pred = model.predict(X_test_transformed)

# Evaluation
print("SGDClassifier")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))

# Predict on a new example from the DataFrame
print("PREDICTIONS USING SGD CLASSIFIER")
print(df.head(3))

data = df['email'][0]
print(data)

# Predict
need_prediction = vectorizer.transform([data])
prediction = model.predict(need_prediction)
print("Prediction for new email:", prediction[0])

# Check if prediction is correct
if prediction[0] == y_encoded[0]:
    print("=======================================================")
    print("CORRECT CLASSIFICATION")
    print("=======================================================")
else:
    print("=======================================================")
    print("WRONG CLASSIFICATION")
    print("=======================================================")
```

```
SGDClassifier
Accuracy: 0.7658333333333334
              precision    recall  f1-score   support

      Change       0.92      0.80      0.86       504
    Incident       0.68      0.93      0.78      1917
     Problem       0.70      0.19      0.30      1007
     Request       0.90      0.94      0.92      1372

    accuracy                           0.77      4800
   macro avg       0.80      0.72      0.71      4800
weighted avg       0.77      0.77      0.73      4800

PREDICTIONS USING SGD CLASSIFIER
       type                                              email
0  Incident  subject unforeseen crash data analysis platfor...
1   Request  subject customer support inquiry seek informat...
2   Request  subject data analytic investment contact reque...
subject unforeseen crash data analysis platform data analysis platform break unexpectedly memory surface small ful
Prediction for new email: 1.0
```

```
=======================================================
CORRECT CLASSIFICATION
=======================================================
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import RidgeClassifier
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Assuming X_train, X_test, y_train, y_test, encoder, df, y_encoded are already defined

# Vectorize the training and test data
vectorizer = TfidfVectorizer()
X_train_transformed = vectorizer.fit_transform(X_train)
X_test_transformed = vectorizer.transform(X_test)

# Train the RidgeClassifier model
model = RidgeClassifier()
model.fit(X_train_transformed, y_train)

# Predict on test data
y_pred = model.predict(X_test_transformed)

# Evaluation
print("RidgeClassifier")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))

# Predict on a new example from the DataFrame
print("PREDICTIONS USING RIDGE CLASSIFIER")
print(df.head(3))

data = df['email'][10]
print(data)

# Predict
need_prediction = vectorizer.transform([data])
prediction = model.predict(need_prediction)
print("Prediction for new email:", prediction[0])

# Check if prediction is correct
if prediction[0] == y_encoded[10]:
    print("=======================================================")
    print("CORRECT CLASSIFICATION")
    print("=======================================================")
else:
    print("=======================================================")
    print("WRONG CLASSIFICATION")
    print("=======================================================")
```

```
RidgeClassifier
Accuracy: 0.7702083333333334
              precision    recall  f1-score   support

      Change       0.91      0.80      0.85       504
    Incident       0.70      0.87      0.78      1917
     Problem       0.61      0.34      0.43      1007
     Request       0.90      0.94      0.92      1372

    accuracy                           0.77      4800
   macro avg       0.78      0.74      0.75      4800
weighted avg       0.76      0.77      0.75      4800

PREDICTIONS USING RIDGE CLASSIFIER
       type                                              email
0  Incident  subject unforeseen crash data analysis platfor...
```

```
1   Request  subject customer support inquiry seek informat...
2   Request  subject data analytic investment contact reque...
subject issue data analytic platform insufficient ram allocation reach report issue data analytic platform platfor
Prediction for new email: 2.0
=====================================================
CORRECT CLASSIFICATION
=====================================================
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Assuming X_train, X_test, y_train, y_test, encoder, df, y_encoded are already defined

# Vectorize the training and test data
vectorizer = TfidfVectorizer()
X_train_transformed = vectorizer.fit_transform(X_train)
X_test_transformed = vectorizer.transform(X_test)

# Train the MLPClassifier model
model = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, random_state=42)
model.fit(X_train_transformed, y_train)

# Predict on test data
y_pred = model.predict(X_test_transformed)

# Evaluation
print("MLPClassifier")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))

# Predict on a new example from the DataFrame
print("PREDICTIONS USING MLP CLASSIFIER")
print(df.head(3))

data = df['email'][10]
print(data)

# Predict
need_prediction = vectorizer.transform([data])
prediction = model.predict(need_prediction)
print("Prediction for new email:", prediction[0])

# Check if prediction is correct
if prediction[0] == y_encoded[10]:
    print("=====================================================")
    print("CORRECT CLASSIFICATION")
    print("=====================================================")
else:
    print("=====================================================")
    print("WRONG CLASSIFICATION")
    print("=====================================================")
```

```
MLPClassifier
Accuracy: 0.79125
              precision    recall  f1-score   support

      Change       0.88      0.86      0.87       504
    Incident       0.78      0.79      0.78      1917
     Problem       0.60      0.56      0.58      1007
     Request       0.91      0.94      0.92      1372

    accuracy                           0.79      4800
   macro avg       0.79      0.79      0.79      4800
weighted avg       0.79      0.79      0.79      4800
```

```
PREDICTIONS USING MLP CLASSIFIER
       type                                            email
0  Incident  subject unforeseen crash data analysis platfor...
1   Request  subject customer support inquiry seek informat...
2   Request  subject data analytic investment contact reque...
subject issue data analytic platform insufficient ram allocation reach report issue data analytic platform platfor
Prediction for new email: 2.0
=====================================================
CORRECT CLASSIFICATION
=====================================================
```

WITH COUNT VECTORIZER

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Assuming X_train, X_test, y_train, y_test, encoder, df, y_encoded are already defined

# Vectorize the training and test data
vectorizer = CountVectorizer()
X_train_transformed = vectorizer.fit_transform(X_train)
X_test_transformed = vectorizer.transform(X_test)

# Train the MultinomialNB model
model = MultinomialNB()
model.fit(X_train_transformed, y_train)

# Predict on test data
y_pred = model.predict(X_test_transformed)

# Evaluation
print("MultinomialNB")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))

# Predict on a new example from the DataFrame
print("PREDICTIONS USING MULTINOMIAL NAIVE BAYES")
print(df.head(3))

data = df['email'][0]
print(data)

# Predict
need_prediction = vectorizer.transform([data])
prediction = model.predict(need_prediction)
print("Prediction for new email:", prediction[0])

# Check if prediction is correct
if prediction[0] == y_encoded[0]:
    print("=====================================================")
    print("CORRECT CLASSIFICATION")
    print("=====================================================")
else:
    print("=====================================================")
    print("WRONG CLASSIFICATION")
    print("=====================================================")
```

```
MultinomialNB
Accuracy: 0.7316666666666667
              precision    recall  f1-score   support
```

```
              Change       0.80      0.75      0.78       504
            Incident       0.72      0.76      0.74      1917
             Problem       0.48      0.45      0.47      1007
             Request       0.91      0.89      0.90      1372

            accuracy                           0.73      4800
           macro avg       0.73      0.71      0.72      4800
        weighted avg       0.73      0.73      0.73      4800

        PREDICTIONS USING MULTINOMIAL NAIVE BAYES
               type                                              email
        0  Incident  subject unforeseen crash data analysis platfor...
        1   Request  subject customer support inquiry seek informat...
        2   Request  subject data analytic investment contact reque...
        subject unforeseen crash data analysis platform data analysis platform break unexpectedly memory surface small ful
        Prediction for new email: 1.0
        ======================================================
        CORRECT CLASSIFICATION
        ======================================================
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Assuming X_train, X_test, y_train, y_test, encoder, df, y_encoded are already defined

# Vectorize the training and test data
vectorizer = CountVectorizer()
X_train_transformed = vectorizer.fit_transform(X_train)
X_test_transformed = vectorizer.transform(X_test)

# Train the LogisticRegression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train_transformed, y_train)

# Predict on test data
y_pred = model.predict(X_test_transformed)

# Evaluation
print("LogisticRegression")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))

# Predict on a new example from the DataFrame
print("PREDICTIONS USING LOGISTIC REGRESSION")
print(df.head(3))

data = df['email'][10]
print(data)

# Predict
need_prediction = vectorizer.transform([data])
prediction = model.predict(need_prediction)
print("Prediction for new email:", prediction[0])

# Check if prediction is correct
if prediction[0] == y_encoded[10]:
    print("======================================================")
    print("CORRECT CLASSIFICATION")
    print("======================================================")
else:
    print("======================================================")
    print("WRONG CLASSIFICATION")
    print("======================================================")
```

```
LogisticRegression
Accuracy: 0.7691666666666667
              precision    recall  f1-score   support

      Change       0.87      0.84      0.86       504
    Incident       0.73      0.81      0.77      1917
     Problem       0.57      0.44      0.49      1007
     Request       0.91      0.93      0.92      1372

    accuracy                           0.77      4800
   macro avg       0.77      0.75      0.76      4800
weighted avg       0.76      0.77      0.76      4800

PREDICTIONS USING LOGISTIC REGRESSION
       type                                                email
0  Incident  subject unforeseen crash data analysis platfor...
1   Request  subject customer support inquiry seek informat...
2   Request  subject data analytic investment contact reque...
subject issue data analytic platform insufficient ram allocation reach report issue data analytic platform platfor
Prediction for new email: 2.0
=====================================================
CORRECT CLASSIFICATION
=====================================================
```

```
from sklearn.pipeline import make_pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report
import joblib  # <- for saving model

# Assuming X_train, y_train, X_test, y_test, and encoder are already defined

# Build and train the model
pipeline = make_pipeline(
    CountVectorizer(),
    MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, random_state=42)
)
pipeline.fit(X_train, y_train)

# Make predictions
y_pred = pipeline.predict(X_test)

# Evaluation
print("MLPClassifier")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))

# Save the pipeline as a .pkl file
joblib.dump(pipeline, "mlp_classifier_pipeline.pkl")
print("Model saved as 'mlp_classifier_pipeline.pkl'")
```

```
MLPClassifier
Accuracy: 0.8125
              precision    recall  f1-score   support

      Change       0.88      0.87      0.87       504
    Incident       0.79      0.82      0.80      1917
     Problem       0.65      0.60      0.62      1007
     Request       0.93      0.94      0.94      1372

    accuracy                           0.81      4800
   macro avg       0.81      0.81      0.81      4800
weighted avg       0.81      0.81      0.81      4800

Model saved as 'mlp_classifier_pipeline.pkl'
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Assuming X_train, X_test, y_train, y_test, encoder, df, y_encoded are already defined

# Vectorize the training and test data
vectorizer = TfidfVectorizer()
X_train_transformed = vectorizer.fit_transform(X_train)
X_test_transformed = vectorizer.transform(X_test)

# Train the SGDClassifier model
model = SGDClassifier()
model.fit(X_train_transformed, y_train)

# Predict on test data
y_pred = model.predict(X_test_transformed)

# Evaluation
print("SGDClassifier")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))

# Predict on a new example from the DataFrame
print("PREDICTIONS USING SGD CLASSIFIER")
print(df.head(3))

data = df['email'][5]
print(data)

# Predict
need_prediction = vectorizer.transform([data])
prediction = model.predict(need_prediction)
print("Prediction for new email:", prediction[0])

# Check if prediction is correct
if prediction[0] == y_encoded[5]:
    print("=======================================================")
    print("CORRECT CLASSIFICATION")
    print("=======================================================")
else:
    print("=======================================================")
    print("WRONG CLASSIFICATION")
    print("=======================================================")
```

```
SGDClassifier
Accuracy: 0.7691666666666667
              precision    recall  f1-score   support

      Change       0.90      0.82      0.86       504
    Incident       0.68      0.93      0.79      1917
     Problem       0.70      0.20      0.32      1007
     Request       0.90      0.94      0.92      1372

    accuracy                           0.77      4800
   macro avg       0.80      0.72      0.72      4800
weighted avg       0.77      0.77      0.73      4800

PREDICTIONS USING SGD CLASSIFIER
       type                                              email
0  Incident  subject unforeseen crash data analysis platfor...
1   Request  subject customer support inquiry seek informat...
2   Request  subject data analytic investment contact reque...
subject concern secure medical datum 2 in-1 convertible laptop norton cvv_no inquire good practice secure medical
Prediction for new email: 3.0
=======================================================
```

```
CORRECT CLASSIFICATION
=====================================================
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Assuming X_train, X_test, y_train, y_test, encoder, df, y_encoded are already defined

# Vectorize the training and test data
vectorizer = CountVectorizer()
X_train_transformed = vectorizer.fit_transform(X_train)
X_test_transformed = vectorizer.transform(X_test)

# Train the DecisionTreeClassifier model
model = DecisionTreeClassifier()
model.fit(X_train_transformed, y_train)

# Predict on test data
y_pred = model.predict(X_test_transformed)

# Evaluation
print("DecisionTreeClassifier")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))

# Predict on a new example from the DataFrame
print("PREDICTIONS USING DECISION TREE")
print(df.head(3))

data = df['email'][10]
print(data)

# Predict
need_prediction = vectorizer.transform([data])
prediction = model.predict(need_prediction)
print("Prediction for new email:", prediction[0])

# Check if prediction is correct
if prediction[0] == y_encoded[10]:
    print("=====================================================")
    print("CORRECT CLASSIFICATION")
    print("=====================================================")
else:
    print("=====================================================")
    print("WRONG CLASSIFICATION")
    print("=====================================================")
```

```
DecisionTreeClassifier
Accuracy: 0.706875
              precision    recall  f1-score   support

      Change       0.74      0.69      0.71       504
    Incident       0.70      0.72      0.71      1917
     Problem       0.48      0.46      0.47      1007
     Request       0.86      0.87      0.87      1372

    accuracy                           0.71      4800
   macro avg       0.70      0.69      0.69      4800
weighted avg       0.70      0.71      0.71      4800

PREDICTIONS USING DECISION TREE
       type                                              email
0  Incident  subject unforeseen crash data analysis platfor...
1   Request  subject customer support inquiry seek informat...
```

```
2   Request  subject data analytic investment contact reque...
subject issue data analytic platform insufficient ram allocation reach report issue data analytic platform platfor
Prediction for new email: 2.0
=======================================================
CORRECT CLASSIFICATION
=======================================================
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Assuming X_train, X_test, y_train, y_test, encoder, df, y_encoded are already defined

# Vectorize the training and test data
vectorizer = CountVectorizer()
X_train_transformed = vectorizer.fit_transform(X_train)
X_test_transformed = vectorizer.transform(X_test)

# Train the GradientBoostingClassifier model
model = GradientBoostingClassifier()
model.fit(X_train_transformed, y_train)

# Predict on test data
y_pred = model.predict(X_test_transformed)

# Evaluation
print("GradientBoostingClassifier")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))

# Predict on a new example from the DataFrame
print("PREDICTIONS USING GRADIENT BOOSTING")
print(df.head(3))

data = df['email'][10]
print(data)

# Predict
need_prediction = vectorizer.transform([data])
prediction = model.predict(need_prediction)
print("Prediction for new email:", prediction[0])

# Check if prediction is correct
if prediction[0] == y_encoded[10]:
    print("=======================================================")
    print("CORRECT CLASSIFICATION")
    print("=======================================================")
else:
    print("=======================================================")
    print("WRONG CLASSIFICATION")
    print("=======================================================")
```

```
GradientBoostingClassifier
Accuracy: 0.7429166666666667
              precision    recall  f1-score   support

      Change       0.90      0.71      0.79       504
    Incident       0.66      0.93      0.77      1917
     Problem       0.65      0.17      0.26      1007
     Request       0.86      0.92      0.89      1372

    accuracy                           0.74      4800
   macro avg       0.77      0.68      0.68      4800
weighted avg       0.74      0.74      0.70      4800
```

```
PREDICTIONS USING GRADIENT BOOSTING
       type                                              email
0  Incident  subject unforeseen crash data analysis platfor...
1   Request  subject customer support inquiry seek informat...
2   Request  subject data analytic investment contact reque...
subject issue data analytic platform insufficient ram allocation reach report issue data analytic platform platfor
Prediction for new email: 1.0
=====================================================
WRONG CLASSIFICATION
=====================================================
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Assuming X_train, X_test, y_train, y_test, encoder, df, y_encoded are already defined

# Vectorize the training and test data
vectorizer = CountVectorizer()
X_train_transformed = vectorizer.fit_transform(X_train)
X_test_transformed = vectorizer.transform(X_test)

# Train the RandomForestClassifier model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_transformed, y_train)

# Predict on test data
y_pred = model.predict(X_test_transformed)

# Evaluation
print("RandomForestClassifier")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))

# Predict on a new example from the DataFrame
print("PREDICTIONS USING RANDOM FOREST")
print(df.head(3))

data = df['email'][10]
print(data)

# Predict
need_prediction = vectorizer.transform([data])
prediction = model.predict(need_prediction)
print("Prediction for new email:", prediction[0])

# Check if prediction is correct
if prediction[0] == y_encoded[10]:
    print("=====================================================")
    print("CORRECT CLASSIFICATION")
    print("=====================================================")
else:
    print("=====================================================")
    print("WRONG CLASSIFICATION")
    print("=====================================================")
```

```
RandomForestClassifier
Accuracy: 0.7897916666666667
              precision    recall  f1-score   support

      Change       0.98      0.72      0.83       504
    Incident       0.69      0.98      0.81      1917
     Problem       0.94      0.25      0.39      1007
     Request       0.90      0.95      0.92      1372
```

```
           accuracy                           0.79      4800
          macro avg       0.88      0.72      0.74      4800
       weighted avg       0.83      0.79      0.76      4800

       PREDICTIONS USING RANDOM FOREST
              type                                              email
       0  Incident  subject unforeseen crash data analysis platfor...
       1   Request  subject customer support inquiry seek informat...
       2   Request  subject data analytic investment contact reque...
       subject issue data analytic platform insufficient ram allocation reach report issue data analytic platform platfor
       Prediction for new email: 2.0
       ========================================================
       CORRECT CLASSIFICATION
       ========================================================
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Assuming X_train, X_test, y_train, y_test, encoder, df, y_encoded are already defined

# Vectorize the training and test data
vectorizer = CountVectorizer()
X_train_transformed = vectorizer.fit_transform(X_train)
X_test_transformed = vectorizer.transform(X_test)

# Train the LinearSVC model
model = LinearSVC()
model.fit(X_train_transformed, y_train)

# Predict on test data
y_pred = model.predict(X_test_transformed)

# Evaluation
print("LinearSVC")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=encoder.categories_[0]))

# Predict on a new example from the DataFrame
print("PREDICTIONS USING LINEAR SVC")
print(df.head(3))

data = df['email'][10]
print(data)

# Predict
need_prediction = vectorizer.transform([data])
prediction = model.predict(need_prediction)
print("Prediction for new email:", prediction[0])

# Check if prediction is correct
if prediction[0] == y_encoded[10]:
    print("========================================================")
    print("CORRECT CLASSIFICATION")
    print("========================================================")
else:
    print("========================================================")
    print("WRONG CLASSIFICATION")
    print("========================================================")
```

```
LinearSVC
Accuracy: 0.7647916666666666
              precision    recall  f1-score   support

      Change       0.86      0.87      0.87       504
```

```
    Incident       0.73      0.79      0.76      1917
     Problem       0.55      0.44      0.49      1007
     Request       0.91      0.93      0.92      1372

    accuracy                           0.76      4800
   macro avg       0.76      0.76      0.76      4800
weighted avg       0.76      0.76      0.76      4800

PREDICTIONS USING LINEAR SVC
       type                                              email
0  Incident  subject unforeseen crash data analysis platfor...
1   Request  subject customer support inquiry seek informat...
2   Request  subject data analytic investment contact reque...
subject issue data analytic platform insufficient ram allocation reach report issue data analytic platform platfor
Prediction for new email: 2.0
=====================================================
CORRECT CLASSIFICATION
=====================================================
```

USING SOME AUTOMATED MACHINE LEARNING FRAMEWORKS

```
# using pycaret
!pip install pycaret
```

```
Collecting pycaret
  Downloading pycaret-3.3.2-py3-none-any.whl.metadata (17 kB)
Requirement already satisfied: ipython>=5.5.0 in /usr/local/lib/python3.11/dist-packages (from pycaret) (7.34.0)
Requirement already satisfied: ipywidgets>=7.6.5 in /usr/local/lib/python3.11/dist-packages (from pycaret) (7.7.1)
Requirement already satisfied: tqdm>=4.62.0 in /usr/local/lib/python3.11/dist-packages (from pycaret) (4.67.1)
Collecting numpy<1.27,>=1.21 (from pycaret)
  Downloading numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (61 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.0/61.0 kB 4.0 MB/s eta 0:00:00
Collecting pandas<2.2.0 (from pycaret)
  Downloading pandas-2.1.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (18 kB)
Requirement already satisfied: jinja2>=3 in /usr/local/lib/python3.11/dist-packages (from pycaret) (3.1.6)
Collecting scipy<=1.11.4,>=1.6.1 (from pycaret)
  Downloading scipy-1.11.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (60 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.4/60.4 kB 3.9 MB/s eta 0:00:00
Collecting joblib<1.4,>=1.2.0 (from pycaret)
  Downloading joblib-1.3.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: scikit-learn>1.4.0 in /usr/local/lib/python3.11/dist-packages (from pycaret) (1.6.1
Collecting pyod>=1.1.3 (from pycaret)
  Downloading pyod-2.0.4.tar.gz (169 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 169.7/169.7 kB 12.0 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: imbalanced-learn>=0.12.0 in /usr/local/lib/python3.11/dist-packages (from pycaret)
Collecting category-encoders>=2.4.0 (from pycaret)
  Downloading category_encoders-2.8.1-py3-none-any.whl.metadata (7.9 kB)
Requirement already satisfied: lightgbm>=3.0.0 in /usr/local/lib/python3.11/dist-packages (from pycaret) (4.5.0)
Requirement already satisfied: numba>=0.55.0 in /usr/local/lib/python3.11/dist-packages (from pycaret) (0.60.0)
Requirement already satisfied: requests>=2.27.1 in /usr/local/lib/python3.11/dist-packages (from pycaret) (2.32.3)
Requirement already satisfied: psutil>=5.9.0 in /usr/local/lib/python3.11/dist-packages (from pycaret) (5.9.5)
Requirement already satisfied: markupsafe>=2.0.1 in /usr/local/lib/python3.11/dist-packages (from pycaret) (3.0.2)
Requirement already satisfied: importlib-metadata>=4.12.0 in /usr/local/lib/python3.11/dist-packages (from pycaret
Requirement already satisfied: nbformat>=4.2.0 in /usr/local/lib/python3.11/dist-packages (from pycaret) (5.10.4)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.11/dist-packages (from pycaret) (3.1.1)
Collecting deprecation>=2.1.0 (from pycaret)
  Downloading deprecation-2.1.0-py2.py3-none-any.whl.metadata (4.6 kB)
Collecting xxhash (from pycaret)
  Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting matplotlib<3.8.0 (from pycaret)
  Downloading matplotlib-3.7.5-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (5.7 kB)
Collecting scikit-plot>=0.3.7 (from pycaret)
  Downloading scikit_plot-0.3.7-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: yellowbrick>=1.4 in /usr/local/lib/python3.11/dist-packages (from pycaret) (1.5)
Requirement already satisfied: plotly>=5.14.0 in /usr/local/lib/python3.11/dist-packages (from pycaret) (5.24.1)
Collecting kaleido>=0.2.1 (from pycaret)
  Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl.metadata (15 kB)
Collecting schemdraw==0.15 (from pycaret)
  Downloading schemdraw-0.15-py3-none-any.whl.metadata (2.2 kB)
Collecting plotly-resampler>=0.8.3.1 (from pycaret)
  Downloading plotly_resampler-0.10.0-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: statsmodels>=0.12.1 in /usr/local/lib/python3.11/dist-packages (from pycaret) (0.14
Collecting sktime==0.26.0 (from pycaret)
  Downloading sktime-0.26.0-py3-none-any.whl.metadata (29 kB)
Collecting tbats>=1.1.3 (from pycaret)
  Downloading tbats-1.1.3-py3-none-any.whl.metadata (3.8 kB)
Collecting pmdarima>=2.0.4 (from pycaret)
  Downloading pmdarima-2.0.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl.meta
Requirement already satisfied: wurlitzer in /usr/local/lib/python3.11/dist-packages (from pycaret) (3.1.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from sktime==0.26.0->pycaret)
Collecting scikit-base<0.8.0 (from sktime==0.26.0->pycaret)
  Downloading scikit_base-0.7.8-py3-none-any.whl.metadata (8.8 kB)
Collecting scikit-learn>1.4.0 (from pycaret)
  Downloading scikit_learn-1.4.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (11 kB)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.11/dist-packages (from category-encoders>=2.
INFO: pip is looking at multiple versions of category-encoders to determine which version is compatible with other
Collecting category-encoders>=2.4.0 (from pycaret)
  Downloading category_encoders-2.8.0-py3-none-any.whl.metadata (7.9 kB)
  Downloading category_encoders-2.7.0-py3-none-any.whl.metadata (7.9 kB)
Requirement already satisfied: sklearn-compat<1,>=0.1 in /usr/local/lib/python3.11/dist-packages (from imbalanced-
Requirement already satisfied: threadpoolctl<4,>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from imbalanced
Requirement already satisfied: zipp>=3.20 in /usr/local/lib/python3.11/dist-packages (from importlib-metadata>=4.1
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.11/dist-packages (from ipython>=5.5.0->p
Collecting jedi>=0.16 (from ipython>=5.5.0->pycaret)
  Downloading jedi-0.19.2-py2.py3-none-any.whl.metadata (22 kB)
```

```
  Downloading jedi-0.19.2-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: decorator in /usr/local/lib/python3.11/dist-packages (from ipython>=5.5.0->pycaret)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.11/dist-packages (from ipython>=5.5.0->pycare
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.11/dist-packages (from ipython>=5.5.0->pyc
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: pygments in /usr/local/lib/python3.11/dist-packages (from ipython>=5.5.0->pycaret)
Requirement already satisfied: backcall in /usr/local/lib/python3.11/dist-packages (from ipython>=5.5.0->pycaret)
Requirement already satisfied: matplotlib-inline in /usr/local/lib/python3.11/dist-packages (from ipython>=5.5.0->
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.11/dist-packages (from ipython>=5.5.0->pycare
Requirement already satisfied: ipykernel>=4.5.1 in /usr/local/lib/python3.11/dist-packages (from ipywidgets>=7.6.5
Requirement already satisfied: ipython-genutils~=0.2.0 in /usr/local/lib/python3.11/dist-packages (from ipywidgets
Requirement already satisfied: widgetsnbextension~=3.6.0 in /usr/local/lib/python3.11/dist-packages (from ipywidge
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from ipywidge
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib<3.8.0-
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib<3.8.0->pyc
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib<3.8.0
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib<3.8.0
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib<3.8.0->py
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib<3.8.0-
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib<3.
Requirement already satisfied: fastjsonschema>=2.15 in /usr/local/lib/python3.11/dist-packages (from nbformat>=4.2
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.11/dist-packages (from nbformat>=4.2.0->p
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in /usr/local/lib/python3.11/dist-packages (from nbforma
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.11/dist-packages (from numba>=
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<2.2.0->pycaret
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.11/dist-packages (from pandas<2.2.0->pycar
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11/dist-packages (from plotly>=5.14.0->py
Collecting dash>=2.9.0 (from plotly-resampler>=0.8.3.1->pycaret)
  Downloading dash-3.0.3-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: orjson<4.0.0,>=3.8.0 in /usr/local/lib/python3.11/dist-packages (from plotly-resamp
Collecting tsdownsample>=0.1.3 (from plotly-resampler>=0.8.3.1->pycaret)
  Downloading tsdownsample-0.1.4.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (8.0 kB)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.11/dist-packages (from p
Requirement already satisfied: urllib3 in /usr/local/lib/python3.11/dist-packages (from pmdarima>=2.0.4->pycaret)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.27.1->pyc
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.27.
Collecting Flask<3.1,>=1.0.4 (from dash>=2.9.0->plotly-resampler>=0.8.3.1->pycaret)
  Downloading flask-3.0.3-py3-none-any.whl.metadata (3.2 kB)
Collecting Werkzeug<3.1 (from dash>=2.9.0->plotly-resampler>=0.8.3.1->pycaret)
  Downloading werkzeug-3.0.6-py3-none-any.whl.metadata (3.7 kB)
Requirement already satisfied: typing-extensions>=4.1.1 in /usr/local/lib/python3.11/dist-packages (from dash>=2.9
Collecting retrying (from dash>=2.9.0->plotly-resampler>=0.8.3.1->pycaret)
  Downloading retrying-1.3.4-py3-none-any.whl.metadata (6.9 kB)
Requirement already satisfied: nest-asyncio in /usr/local/lib/python3.11/dist-packages (from dash>=2.9.0->plotly-r
Requirement already satisfied: debugpy>=1.0 in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipy
Requirement already satisfied: jupyter-client>=6.1.12 in /usr/local/lib/python3.11/dist-packages (from ipykernel>=
Requirement already satisfied: pyzmq>=17 in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywid
Requirement already satisfied: tornado>=6.1 in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipy
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /usr/local/lib/python3.11/dist-packages (from jedi>=0.16->ip
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=2.6->nbf
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.11/dist-packages (fr
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=2.
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=2.6->nb
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.11/dist-packages (from jupyter-core!=5.
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.11/dist-packages (from pexpect>4.3->ipyth
Requirement already satisfied: wcwidth in /usr/local/lib/python3.11/dist-packages (from prompt-toolkit!=3.0.0,!=3.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->mat
Requirement already satisfied: notebook>=4.4.1 in /usr/local/lib/python3.11/dist-packages (from widgetsnbextension
Requirement already satisfied: itsdangerous>=2.1.2 in /usr/local/lib/python3.11/dist-packages (from Flask<3.1,>=1.
Requirement already satisfied: click>=8.1.3 in /usr/local/lib/python3.11/dist-packages (from Flask<3.1,>=1.0.4->da
Requirement already satisfied: blinker>=1.6.2 in /usr/local/lib/python3.11/dist-packages (from Flask<3.1,>=1.0.4->
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widge
Requirement already satisfied: nbconvert>=5 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widg
Requirement already satisfied: Send2Trash>=1.8.0 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1-
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->
Requirement already satisfied: prometheus-client in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1-
Requirement already satisfied: nbclassic>=0.4.7 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->
Requirement already satisfied: notebook-shim>=0.2.3 in /usr/local/lib/python3.11/dist-packages (from nbclassic>=0.
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->noteb
Requirement already satisfied: bleach!=5.0.0 in /usr/local/lib/python3.11/dist-packages (from bleach[css]!=5.0.0->
Requirement already satisfied: defusedxml in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->
```

```
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->
Requirement already satisfied: mistune<4,>=2.0.3 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->no
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->note
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5-
Requirement already satisfied: argon2-cffi-bindings in /usr/local/lib/python3.11/dist-packages (from argon2-cffi->
Requirement already satisfied: webencodings in /usr/local/lib/python3.11/dist-packages (from bleach!=5.0.0->bleach
Requirement already satisfied: tinycss2<1.5,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from bleach[css]!=
Requirement already satisfied: jupyter-server<3,>=1.8 in /usr/local/lib/python3.11/dist-packages (from notebook-sh
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from argon2-cffi-bindings->
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4->nbco
Requirement already satisfied: pycparser in /usr/local/lib/python3.11/dist-packages (from cffi>=1.0.1->argon2-cffi
Requirement already satisfied: anyio>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from jupyter-server<3,>=1.
Requirement already satisfied: websocket-client in /usr/local/lib/python3.11/dist-packages (from jupyter-server<3,
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio>=3.1.0->jupyter
Downloading pycaret-3.3.2-py3-none-any.whl (486 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 486.1/486.1 kB 22.9 MB/s eta 0:00:00
Downloading schemdraw-0.15-py3-none-any.whl (106 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 106.8/106.8 kB 8.2 MB/s eta 0:00:00
Downloading sktime-0.26.0-py3-none-any.whl (21.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.8/21.8 MB 75.2 MB/s eta 0:00:00
Downloading category_encoders-2.7.0-py3-none-any.whl (85 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.4/85.4 kB 7.5 MB/s eta 0:00:00
Downloading deprecation-2.1.0-py2.py3-none-any.whl (11 kB)
Downloading joblib-1.3.2-py3-none-any.whl (302 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 302.2/302.2 kB 20.6 MB/s eta 0:00:00
Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl (79.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.9/79.9 MB 9.5 MB/s eta 0:00:00
Downloading matplotlib-3.7.5-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.6/11.6 MB 100.6 MB/s eta 0:00:00
Downloading numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (18.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.3/18.3 MB 92.2 MB/s eta 0:00:00
Downloading pandas-2.1.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.2/12.2 MB 104.8 MB/s eta 0:00:00
Downloading plotly_resampler-0.10.0-py3-none-any.whl (80 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 80.7/80.7 kB 6.5 MB/s eta 0:00:00
Downloading pmdarima-2.0.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (2.2 M
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.2/2.2 MB 71.1 MB/s eta 0:00:00
Downloading scikit_learn-1.4.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.1/12.1 MB 101.4 MB/s eta 0:00:00
Downloading scikit_plot-0.3.7-py3-none-any.whl (33 kB)
Downloading scipy-1.11.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (36.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 36.4/36.4 MB 19.9 MB/s eta 0:00:00
Downloading tbats-1.1.3-py3-none-any.whl (44 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.0/44.0 kB 3.2 MB/s eta 0:00:00
Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.8/194.8 kB 14.8 MB/s eta 0:00:00
Downloading dash-3.0.3-py3-none-any.whl (8.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.0/8.0 MB 98.3 MB/s eta 0:00:00
Downloading jedi-0.19.2-py2.py3-none-any.whl (1.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 60.5 MB/s eta 0:00:00
Downloading scikit_base-0.7.8-py3-none-any.whl (130 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 130.1/130.1 kB 9.7 MB/s eta 0:00:00
Downloading tsdownsample-0.1.4.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 60.0 MB/s eta 0:00:00
Downloading flask-3.0.3-py3-none-any.whl (101 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 101.7/101.7 kB 8.1 MB/s eta 0:00:00
Downloading werkzeug-3.0.6-py3-none-any.whl (227 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 228.0/228.0 kB 17.0 MB/s eta 0:00:00
Downloading retrying-1.3.4-py3-none-any.whl (11 kB)
Building wheels for collected packages: pyod
  Building wheel for pyod (setup.py) ... done
  Created wheel for pyod: filename=pyod-2.0.4-py3-none-any.whl size=200479 sha256=1da9e61f7a46956c918c20fcf5f46387
  Stored in directory: /root/.cache/pip/wheels/cb/54/28/d02f62720600bc815c41219efedd0cb86889737fb5ea3c8f9a
Successfully built pyod
Installing collected packages: kaleido, xxhash, Werkzeug, scikit-base, schemdraw, retrying, numpy, joblib, jedi, d
  Attempting uninstall: Werkzeug
    Found existing installation: Werkzeug 3.1.3
    Uninstalling Werkzeug-3.1.3:
      Successfully uninstalled Werkzeug-3.1.3
  Attempting uninstall: numpy
    Found existing installation: numpy 2.0.2
    Uninstalling numpy-2.0.2:
      Successfully uninstalled numpy-2.0.2
```