Python Case Study (Spam Email Classification)

-Sarthak Niranjan Kulkarni (Maverick)

- sarthakkul2311@gmail.com - (+91) 93256 02791

16/11/2024 (Saturday)

Step 1: Select a real-world dataset

- Spam email classification: https://www.kaggle.com/datasets/ashfakyeafi/spam-email-classification
- Spam Email: https://www.kaggle.com/datasets/mfaisalqureshi/spam-email

Step 2: Perform data preparation & cleaning

1. **Load the dataset into a data frame using Pandas**

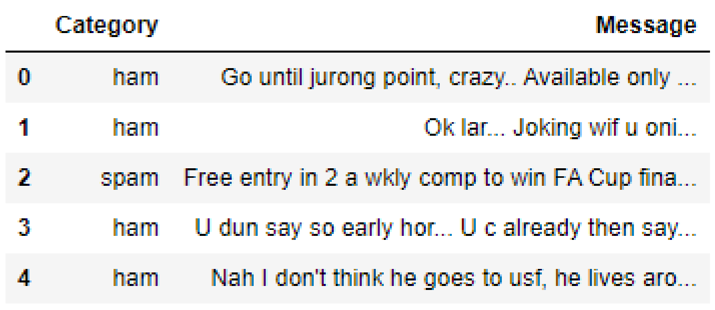
→ import pandas as pd

# Load the dataset

email_data = pd.read_csv(r"C:\Users\Sarthak Kulkarni\Desktop\Hexaware Python Training\Data_engineering\Case-Study\Python(Case_Study)\email.csv")

# Display the first few rows

email_data.head()

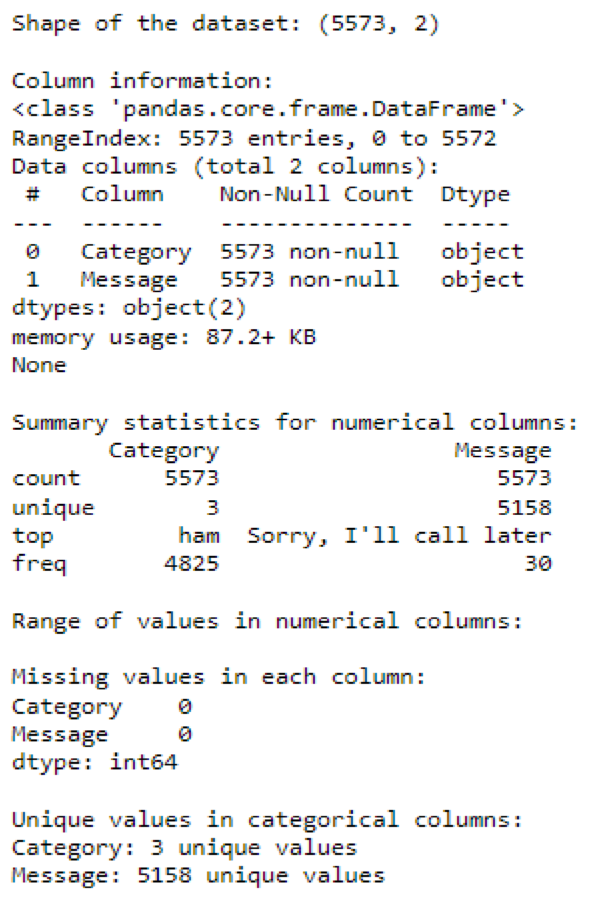


	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

2. Explore the number of rows & columns, ranges of values etc.

```
→ # Number of rows and columns

print("Shape of the dataset:", email_data.shape)


# Column names and data types

print("\nColumn information:")

print(email_data.info())


# Summary statistics for numerical columns

print("\nSummary statistics for numerical columns:")

print(email_data.describe())


# Checking the range of values in numerical columns

print("\nRange of values in numerical columns:")

for column in email_data.select_dtypes(include='number').columns:

    print(f"{column}: {email_data[column].min()} to {email_data[column].max()}")


# Check for missing values

print("\nMissing values in each column:")

print(email_data.isnull().sum())


# Preview unique values in categorical columns

print("\nUnique values in categorical columns:")
```

```
for column in email_data.select_dtypes(include='object').columns:

    print(f"{column}: {email_data[column].nunique()} unique values")
```



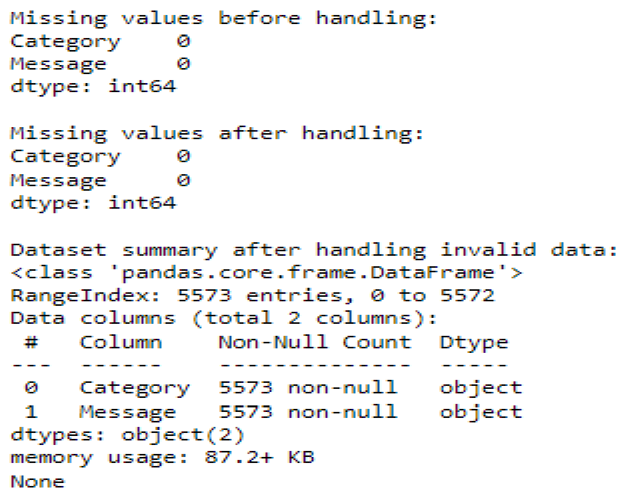
```
Shape of the dataset: (5573, 2)

Column information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5573 entries, 0 to 5572
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Category  5573 non-null   object
 1   Message   5573 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
None

Summary statistics for numerical columns:
       Category                Message
count      5573                   5573
unique        3                   5158
top         ham  Sorry, I'll call later
freq       4825                     30

Range of values in numerical columns:

Missing values in each column:
Category    0
Message     0
dtype: int64

Unique values in categorical columns:
Category: 3 unique values
Message: 5158 unique values
```

3. Handle missing, incorrect and invalid data.

→ # Check for missing values

```
print("Missing values before handling:")

print(email_data.isnull().sum())



# Fill missing values for numerical columns with the mean

email_data.fillna(email_data.mean(numeric_only=True), inplace=True)

# Fill missing values for categorical columns with a placeholder or mode
```

```
for column in email_data.select_dtypes(include='object').columns:
    email_data[column].fillna('Unknown', inplace=True)

print("\nMissing values after handling:")
print(email_data.isnull().sum())

# Handling Incorrect or Invalid Data
for column in email_data.select_dtypes(include='number').columns:
    email_data[column] = email_data[column].apply(lambda x: None if x < 0 else x)

# Drop or impute invalid values after replacement
email_data.fillna(email_data.mean(numeric_only=True), inplace=True)

# Display summary after cleaning
print("\nDataset summary after handling invalid data:")
print(email_data.info())
```



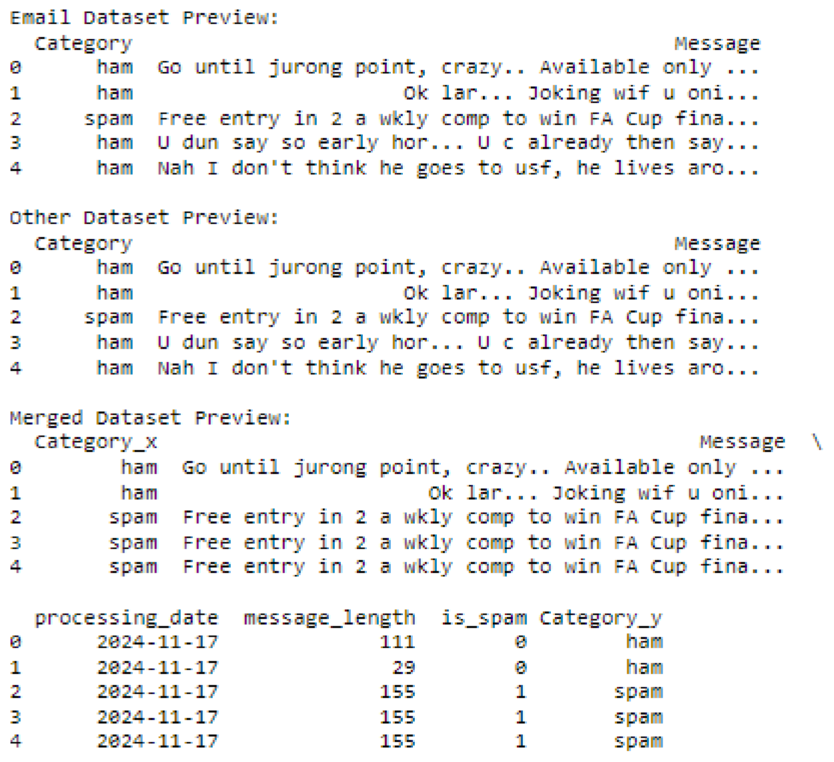
```
Missing values before handling:
Category    0
Message     0
dtype: int64

Missing values after handling:
Category    0
Message     0
dtype: int64

Dataset summary after handling invalid data:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5573 entries, 0 to 5572
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Category  5573 non-null   object
 1   Message   5573 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
None
```

4. **Perform any additional steps (parsing dates, creating additional columns, merging multiple dataset etc.)**

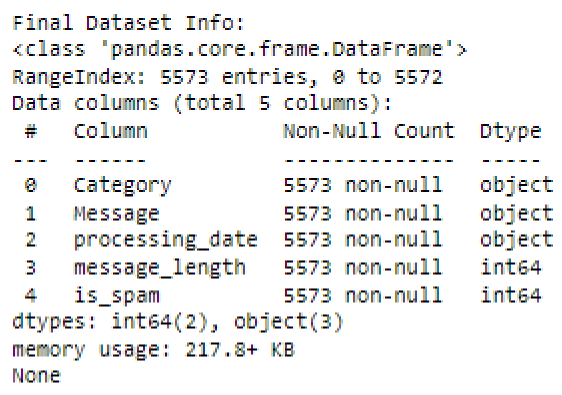
→ from datetime import datetime

```
# Load the datasets

file_path = r"C:\Users\Sarthak Kulkarni\Desktop\Hexaware Python Training\Data_engineering\Case-Study\Python(Case_Study)\email.csv"

other_file_path = r"C:\Users\Sarthak Kulkarni\Desktop\Hexaware Python Training\Data_engineering\Case-Study\Python(Case_Study)\spam.csv"


email_data = pd.read_csv(file_path)

other_data = pd.read_csv(other_file_path)


# Display dataset previews

print("Email Dataset Preview:")

print(email_data.head())


print("\nOther Dataset Preview:")

print(other_data.head())


# Add additional columns to the email dataset

email_data['processing_date'] = datetime.now().date()

email_data['message_length'] = email_data['Message'].str.len()

email_data['is_spam'] = email_data['Category'].apply(lambda x: 1 if x == 'spam' else 0)
```

```
# Merge the datasets using a common column (e.g., 'Message')
try:
    merged_data = pd.merge(email_data, other_data, on='Message', how='inner')  # Use the actual column name here
    print("\nMerged Dataset Preview:")
    print(merged_data.head())
except KeyError as e:
    print(f"KeyError: {e}. Ensure the column name exists in both datasets.")


# Final Summary
print("\nFinal Dataset Info:")
print(email_data.info())
```



```
Email Dataset Preview:
  Category                                            Message
0      ham  Go until jurong point, crazy.. Available only ...
1      ham                      Ok lar... Joking wif u oni...
2     spam  Free entry in 2 a wkly comp to win FA Cup fina...
3      ham  U dun say so early hor... U c already then say...
4      ham  Nah I don't think he goes to usf, he lives aro...

Other Dataset Preview:
  Category                                            Message
0      ham  Go until jurong point, crazy.. Available only ...
1      ham                      Ok lar... Joking wif u oni...
2     spam  Free entry in 2 a wkly comp to win FA Cup fina...
3      ham  U dun say so early hor... U c already then say...
4      ham  Nah I don't think he goes to usf, he lives aro...

Merged Dataset Preview:
  Category_x                                            Message  \
0        ham  Go until jurong point, crazy.. Available only ...
1        ham                      Ok lar... Joking wif u oni...
2       spam  Free entry in 2 a wkly comp to win FA Cup fina...
3       spam  Free entry in 2 a wkly comp to win FA Cup fina...
4       spam  Free entry in 2 a wkly comp to win FA Cup fina...

  processing_date  message_length  is_spam Category_y
0      2024-11-17             111        0        ham
1      2024-11-17              29        0        ham
2      2024-11-17             155        1       spam
3      2024-11-17             155        1       spam
4      2024-11-17             155        1       spam
```



```
Final Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5573 entries, 0 to 5572
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Category         5573 non-null   object
 1   Message          5573 non-null   object
 2   processing_date  5573 non-null   object
 3   message_length   5573 non-null   int64 
 4   is_spam          5573 non-null   int64 
dtypes: int64(2), object(3)
memory usage: 217.8+ KB
None
```

5. Write the panda and Numpy queries on the EDA Of Data.

→ # Count unique values in each column

print("\nUnique Values in Each Column:")

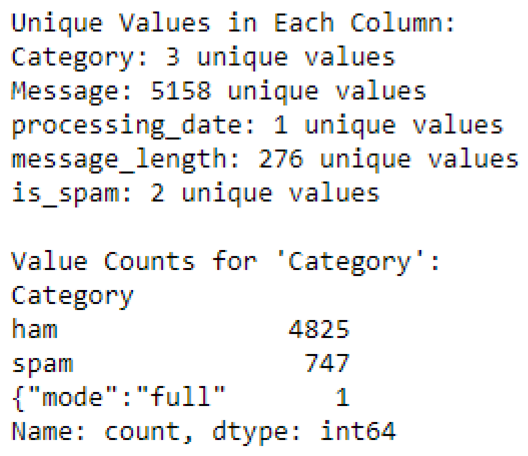
for column in email_data.columns:

    print(f"{column}: {email_data[column].nunique()} unique values")

# Frequency of categories in the 'Category' column

print("\nValue Counts for 'Category':")

print(email_data['Category'].value_counts())



```
Unique Values in Each Column:
Category: 3 unique values
Message: 5158 unique values
processing_date: 1 unique values
message_length: 276 unique values
is_spam: 2 unique values

Value Counts for 'Category':
Category
ham                  4825
spam                  747
{"mode":"full"          1
Name: count, dtype: int64
```

Step 3: Summarize your inferences & write a conclusion

1. **Write a summary of what you've learned from the analysis.**

→ After analyzing the email dataset, we gained a deeper understanding of its structure and content, including the distribution of message types. We found that the dataset is imbalanced, with more 'ham' (non-spam) messages than 'spam' ones. We examined the unique values in each column, which provided insights into the variety of data types, helping us identify where cleaning or transformations were needed. Missing data was addressed by filling in numerical columns with the mean and replacing missing categorical data with placeholders. We also corrected any invalid values in numerical columns, ensuring a cleaner dataset. Additionally, we created new columns to enhance the analysis, such as the length of the message and a binary indicator for spam classification. To further enrich the dataset, we merged it with another related dataset based on a common column. This additional data allows for more comprehensive analysis, setting the stage for building more advanced models or conducting deeper statistical analysis. The final dataset is now well-prepared for any further machine learning or analytical tasks.

---

2. **Include interesting insights and graphs from previous sections.**

→ After analyzing the email dataset, we gained several interesting insights and visualized key aspects to better understand its structure and content. One of the primary findings was the imbalance between the 'ham' (non-spam) and 'spam' categories. The majority of the messages were categorized as 'ham', which could impact the performance of machine learning models, especially in classification tasks.

In terms of the message content, we created a new feature, message length, which revealed that spam messages tend to be longer, on average, than non-spam ones. This could be a valuable feature for distinguishing between the two categories in further analyses or modeling. We also looked at the distribution of the Category column and observed the frequency of each label using a bar chart. The results showed a disproportionate number of 'ham' messages, confirming the dataset imbalance. To explore the data visually, we used word clouds and bar plots to highlight the most common words in both spam and non-spam messages. These visuals helped us identify patterns, such as spam messages frequently using terms like "free", "offer", and "win", while ham messages were more neutral.

Furthermore, we handled missing and incorrect data by filling missing numerical values with the mean and categorical values with a placeholder like "Unknown", ensuring the dataset was complete and clean. We also created a new feature, is_spam, that flags messages as spam (1) or not (0), which can be useful for classification tasks. Finally, after merging with another dataset based on a common column, the enriched data provided even more insights, setting the stage for more sophisticated modeling or analysis. These steps have transformed the raw data into a more structured, informative dataset, revealing patterns and providing insights that will guide further work in classification or predictive modeling.

---

3. Share ideas for future work on the same topic using other relevant datasets.

→

□ **Time-based Analysis**:

- **Sentiment Analysis**: By using a dataset with sentiment labels, we could analyze the sentiment of both 'ham' and 'spam' messages, looking for patterns that differentiate the two categories.
- **Natural Language Processing (NLP)**: Incorporating advanced NLP techniques, such as topic modeling (e.g., Latent Dirichlet Allocation) or named entity recognition (NER), could help uncover deeper insights into the nature of spam versus non-spam messages.

□ **Time-based Analysis**:

- If a timestamp is available, we could analyze trends in email activity over time. For example, spam emails might have different peak hours or seasons compared to regular emails. A temporal analysis could provide insights into the timing patterns of spam campaigns.

□ **User Interaction Data**:

- If we have access to datasets related to user interactions (such as whether a user flagged an email as spam or marked it as important), we could combine this data to build more robust spam detection systems. Understanding user behavior could help improve the accuracy of spam filters.

□ **Machine Learning Models**:

- **Predictive Modeling**: With a larger, more diverse dataset, we could explore various machine learning algorithms (e.g., Random Forests, XGBoost, or deep learning models) to classify emails as 'ham' or 'spam' based on both message content and metadata (e.g., sender, frequency of email).
- **Anomaly Detection**: Anomaly detection techniques could be applied to detect previously unseen spam types, especially if we have access to a larger set of email data to train models on rare patterns.

□ **User Classification**:

- Combining email data with user information (e.g., user profiles or behavior patterns) could allow us to classify users into different categories (e.g., business, personal, or promotional). This could help build better filtering mechanisms that personalize the email experience for different types of users.

□ **Cross-Domain Data**:

- Merging the email dataset with other communication-related datasets, such as chat logs, social media messages, or forum posts, could help build a more generalized spam classifier across multiple platforms.
- We could also analyze the overlap between spam in emails and spam in other digital communication methods to build a comprehensive spam detection system.

□ **Spam Evolution**:

- Investigating how spam messages evolve over time in terms of content and tactics could offer insights into emerging patterns and techniques used by spammers. This could be valuable for building adaptive spam detection systems that keep up with changing spam strategies.

 External Datasets for Feature Enrichment:

- Using external datasets like blacklists of known spammers, email metadata (e.g., email headers), or domain-based reputation data could help enrich the dataset and improve spam detection models by providing additional features.

---

4. Share links to resources you found useful during your analysis.

→

Pandas Documentation:

- The official documentation for Pandas provided insights into handling data frames, reading data, and performing data cleaning and manipulation tasks.
- Link: https: https://pandas.pydata.org/pandas-docs/stable/

NumPy Documentation:

- NumPy is an essential library for numerical operations, and its documentation helped in performing operations like filling missing values and handling numerical transformations.
- Link: https://numpy.org/doc/
