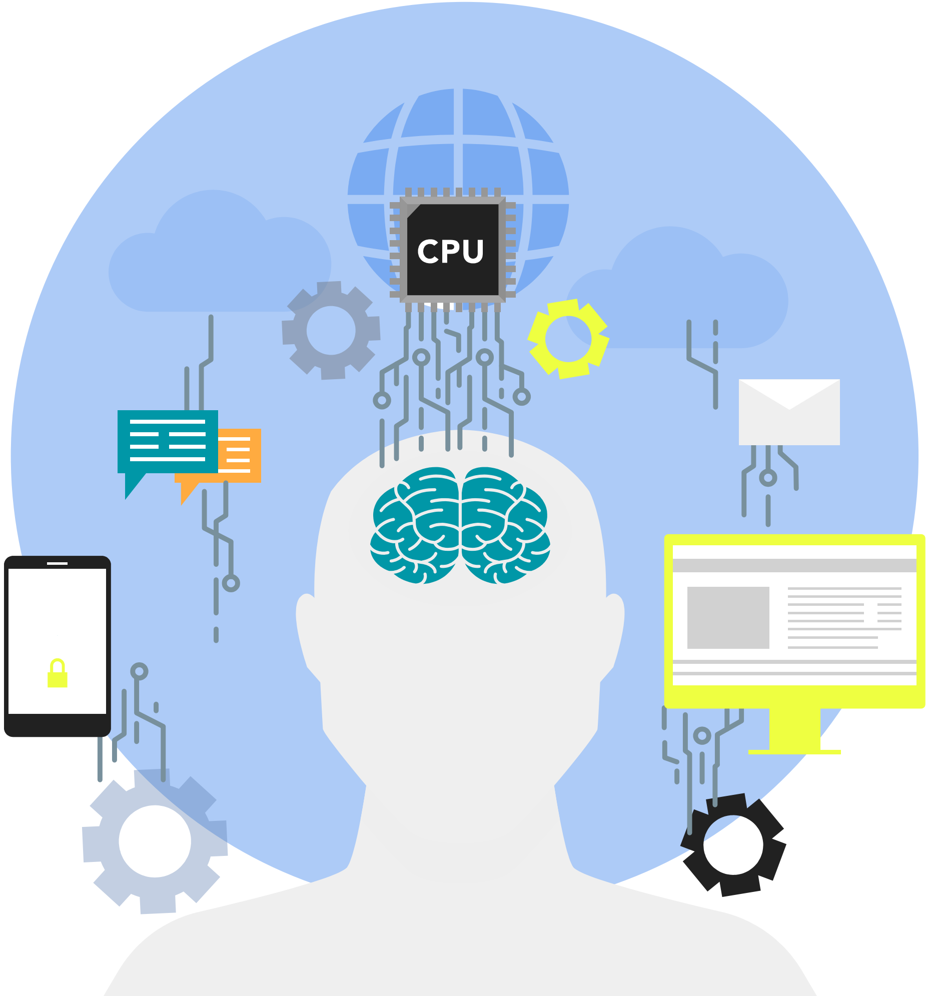


Duplicate Questions Pair

Natural Language Processing

Team Members

Mentor : Dr. Sujoy Das Sir

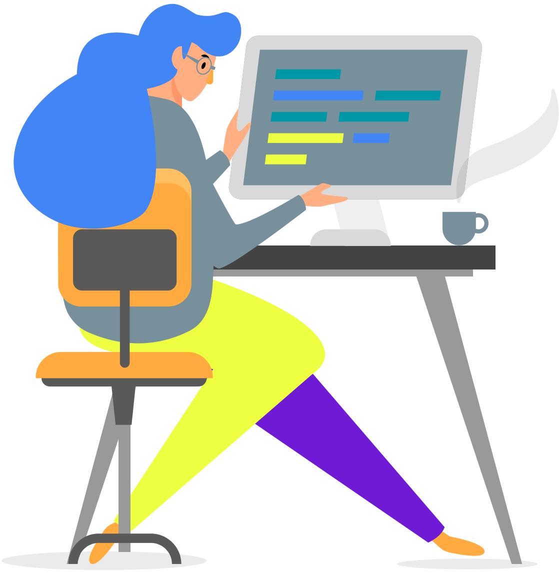
01 **Ambuj Kumar Rai**

212120141

02 **Nitin Kapoor**

212120146

Table of Contents



01 Problem Statement and Objective

02 Prerequisites for Project Development

03 Dataset Description

04 Project Development Overflow

05 Basic Data Analysis

06 Feature Engineering

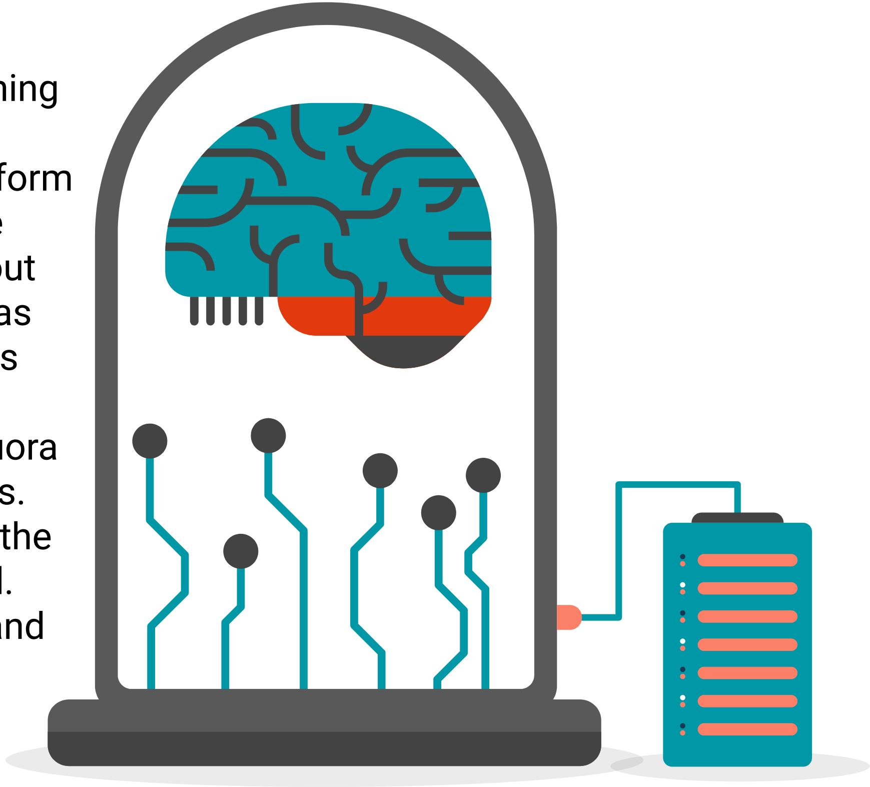
Problem Statement

Imagine you're searching for information online, and you come across many articles and FAQs that essentially say the same thing but in different ways. This can be frustrating and confusing.

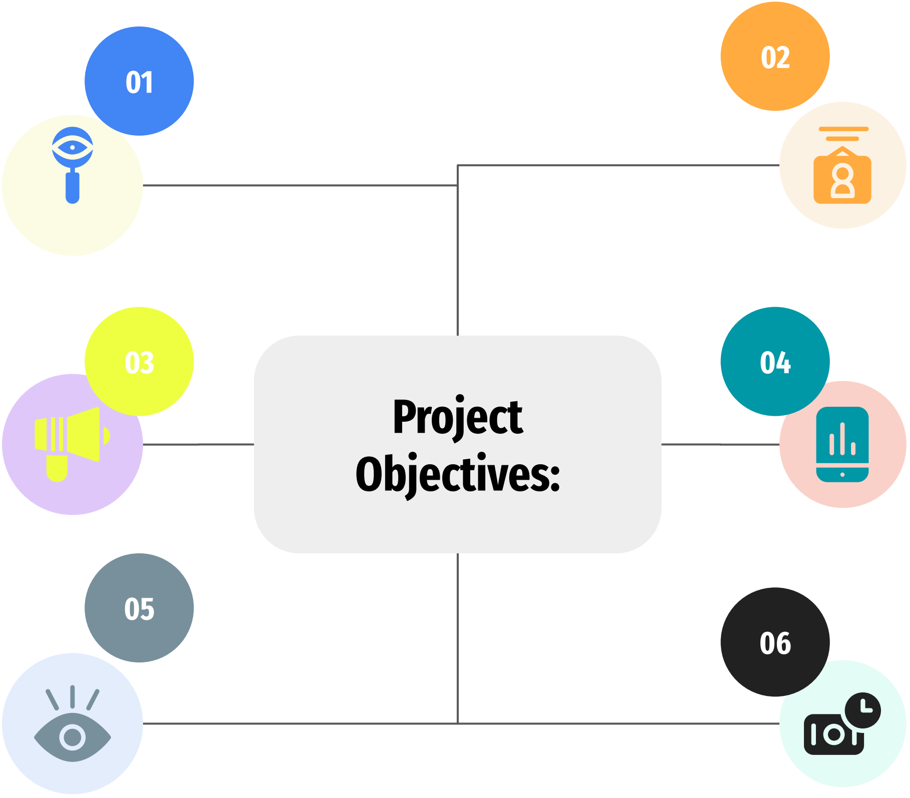
Our specific problem revolves around Quora, a well-known platform where people from around the world ask questions and receive answers from a diverse community. Whether it's questions about science, technology, history, or personal development, Quora has become a valuable resource for knowledge seekers and experts alike.

To tackle this issue of information repetition and confusion, Quora employs a system that lets users vote on the quality of answers. This means that the best and most accurate responses rise to the top, making it easier for users to find the information they need.

In essence, our goal is to make online information more clear and helpful, especially on platforms like Quora where people seek answers to their questions.



Objective



Project Objectives:

Automated Similarity Detection:

Develop an NLP model to automate the identification of similar queries.

Improve Search Relevance:

Enhance the relevance of search results by minimizing repetitive content.

Enhance User Engagement:

Increase user engagement by providing clearer and more concise responses.

Boost Platform Credibility:

Elevate the platform's credibility by ensuring high-quality, non-repetitive answers.

Reduce Moderator Workload:

Reduce the workload on human moderators by automating the duplicate question detection process.

Optimize User Experience:

Optimize the overall user experience by eliminating confusion caused by redundant content.

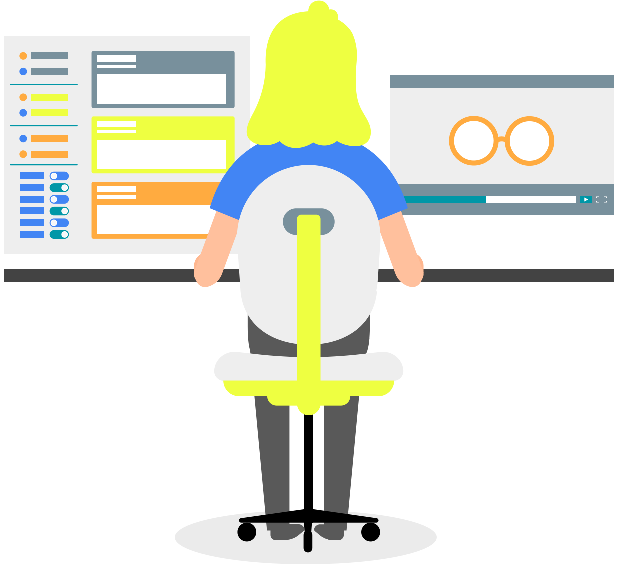
Prerequisites for Project Development

01

Python

02

Matplotlib



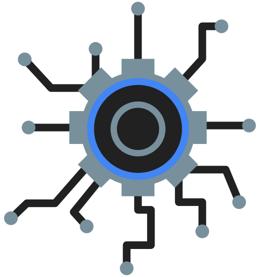
03

Pandas

04

Sklearn

Dataset Description



❑ Quora hosted a popular dataset on Kaggle for a competition.

❑ The dataset consists of five columns.

❑ Two columns contain pairs of different questions.

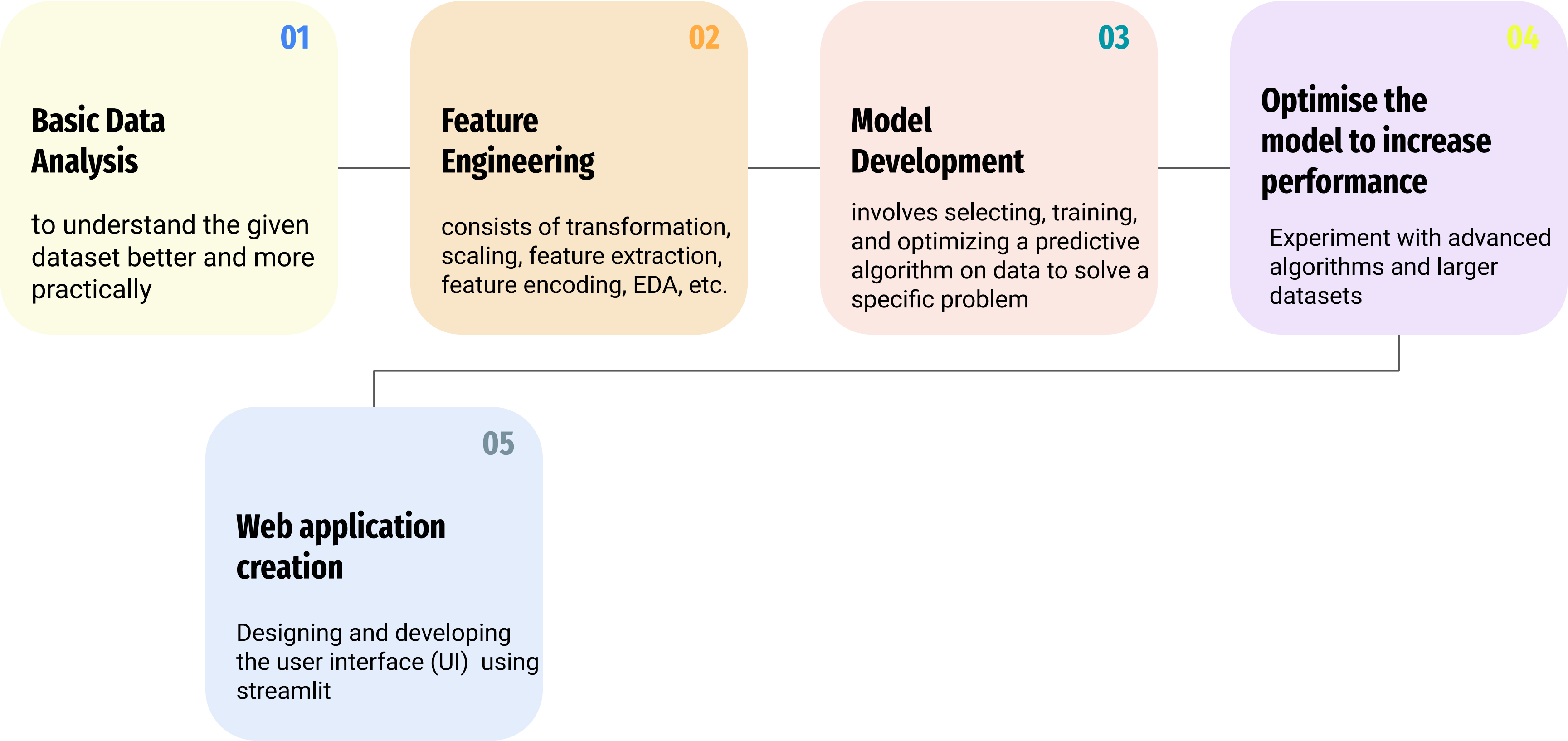
❑ Two additional columns contain question IDs associated with the respective questions.

❑ The last column serves as the target variable, which is in binary format.

❑ In this binary format, a value of 1 indicates that the questions are duplicates.

❑ Conversely, a value of 0 signifies that the questions are not duplicates.

Project Development Overflow



01

Basic Data Analysis

to understand the given dataset better and more practically

02

Feature Engineering

consists of transformation, scaling, feature extraction, feature encoding, EDA, etc.

03

Model Development

involves selecting, training, and optimizing a predictive algorithm on data to solve a specific problem

04

Optimise the model to increase performance

Experiment with advanced algorithms and larger datasets

05

Web application creation

Designing and developing the user interface (UI) using streamlit

Basic Data Analysis

Load the Dataset

import libraries, load the data, and take a sample if needed.

Check Nulls:

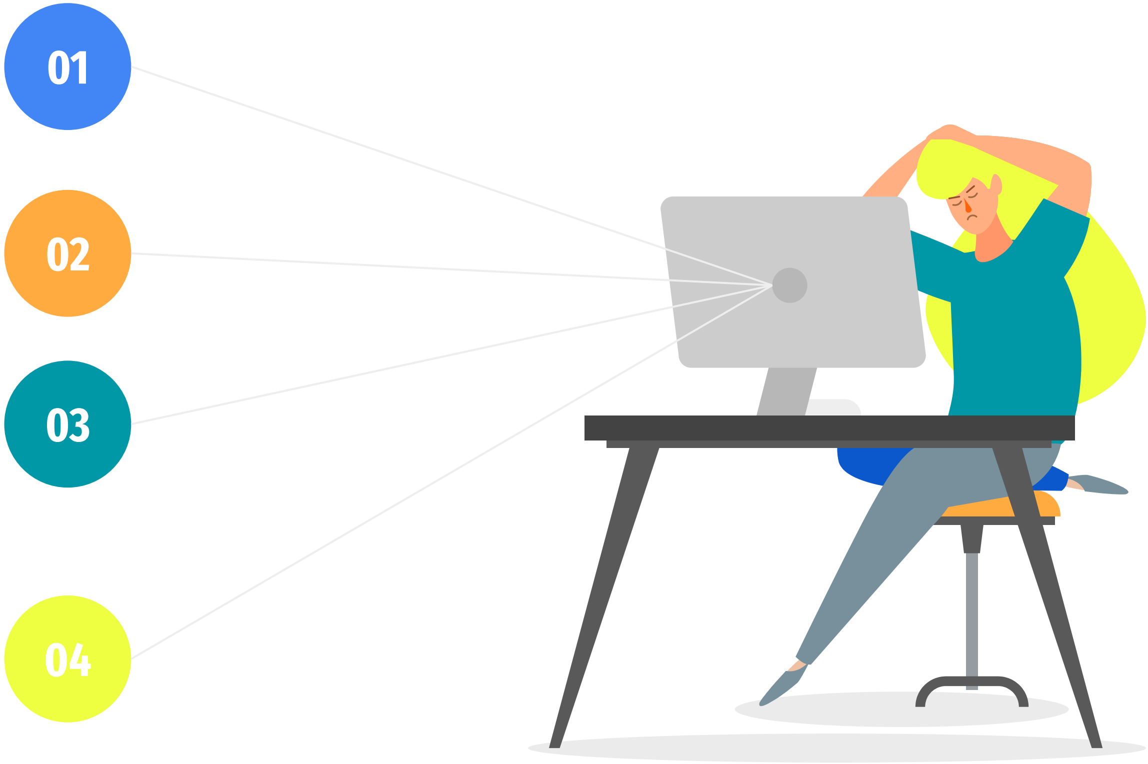
Examine missing values in the dataset.

Check Duplicates:

Identify duplicate entries and assess class balance.

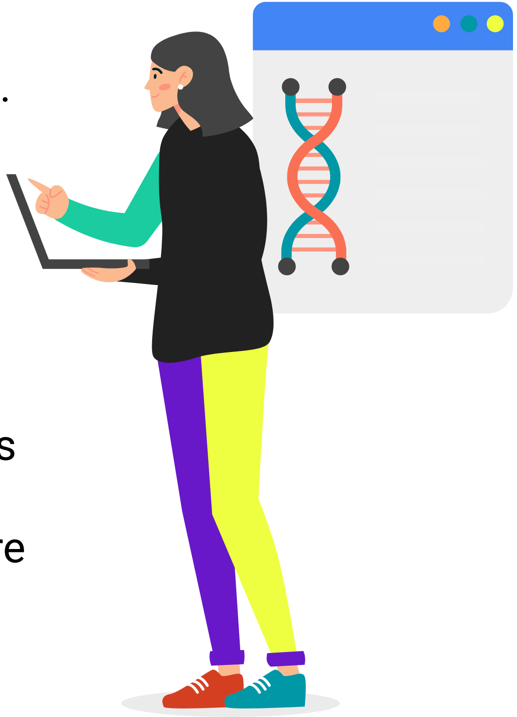
Check Repeated Questions:

Count unique and repeated questions, visualize using a histogram.



Featuring Engineering

- ❑ Feature engineering is a classic technique for enhancing a model's accuracy by adding new features.
- ❑ Crucial features can directly impact model performance.
- ❑ Feature engineering encompasses transformations, scaling, feature extraction, and encoding.
- ❑ In this context, 7 new features will be added to the existing dataset.
- ❑ The Bag of Words model will generate distinct features for questions 1 and 2.
- ❑ These features will undergo exploratory analysis before being used as inputs for machine learning models.



1. Question Length

The size of the question is a critical feature because when we vectorize it, the question gets split by words, so having the length feature is good. The length we are having is the character-wise length. So it will create 2 new features for the length of questions 1 and 2.

```
new_df['q1_len'] = new_df['question1'].str.len()
new_df['q2_len'] = new_df['question2'].str.len()
```

2. Number of Words

The number of words in both questions is another feature that should impact the model performance. So, it will add 2 new features for questions 1 and 2. To add the feature, split the sentence with space and extract the length of the list.

```
new_df['q1_num_words'] = new_df['question1'].apply(lambda row: len(row.split(" ")))
new_df['q2_num_words'] = new_df['question2'].apply(lambda row: len(row.split(" ")))
new_df.head()
```

3. Common Words

Another feature is to know how many common words there are in both questions. It helps identify the similarity between both questions. Calculating where you only need to apply the intersection between both questions is simple. For this, we find the number of unique words in both questions and apply the set intersection to the set length.

```
def common_words(row):
    w1 = set(map(lambda word: word.lower().strip(), row['question1'].split(" ")))
    w2 = set(map(lambda word: word.lower().strip(), row['question2'].split(" ")))
    return len(w1 & w2)
new_df['word_common'] = new_df.apply(common_words, axis=1)
new_df.head()
```

4. Total Words

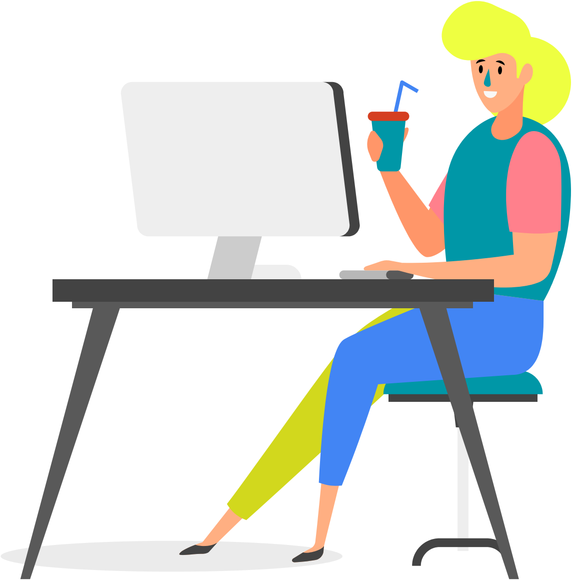
The sum of the total number of unique words in each question. In simple terms, find the number of unique words in both questions and return their sum.

```
def total_words(row):
    w1 = set(map(lambda word: word.lower().strip(), row['question1'].split(" ")))
    w2 = set(map(lambda word: word.lower().strip(), row['question2'].split(" ")))
    return (len(w1) + len(w2))
new_df['word_total'] = new_df.apply(total_words, axis=1)
new_df.head()
```

5. Words Share

It is one exciting feature and simple to add. To calculate, divide the common words by the total number of words..

```
new_df['word_share'] = round(new_df['word_common']/new_df['word_total'],2)
new_df.head()
```



Machine Learning Model Creation

# Import necessary libraries

# Step 1: Separate Independent and Dependent Features

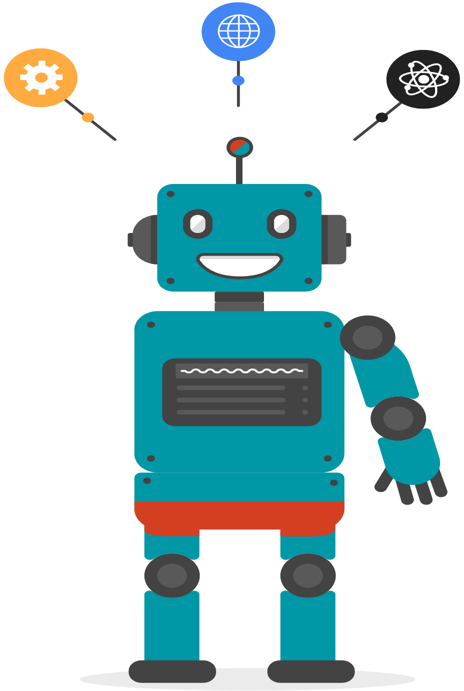
```
ques_df = new_df[['question1', 'question2']]
final_df = new_df.drop(columns=['id', 'qid1', 'qid2', 'question1', 'question2'])
```

# Step 2: Vectorizing the Features

```
from sklearn.feature_extraction.text import CountVectorizer
questions = list(ques_df['question1']) + list(ques_df['question2'])
cv = CountVectorizer(max_features=3000)
q1_arr, q2_arr = np.vsplit(cv.fit_transform(questions).toarray(), 2)
temp_df1 = pd.DataFrame(q1_arr, index=ques_df.index)
temp_df2 = pd.DataFrame(q2_arr, index=ques_df.index)
temp_df = pd.concat([temp_df1, temp_df2], axis=1)
final_df = pd.concat([final_df, temp_df], axis=1)
```

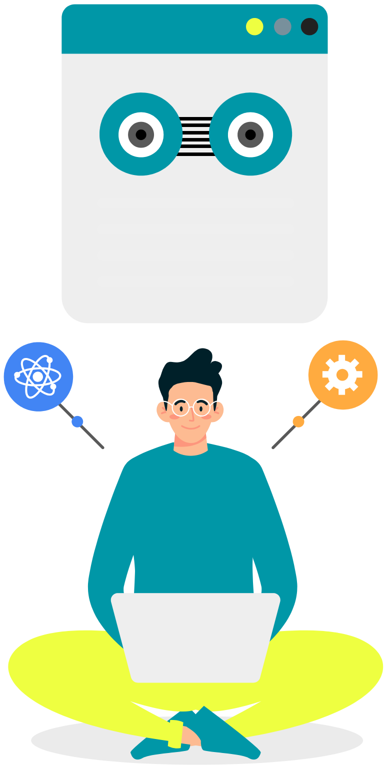
# Step 3: Train-Test Split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(final_df.iloc[:, 1:].values, final_df.iloc[:, 0].values, test_size=0.2, random_state=1)
```



Machine Learning Model Creation

```
# Step 4: Train the Machine Learning Models
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

xgb = XGBClassifier()
xgb.fit(X_train, y_train)
y_pred_xgb = xgb.predict(X_test)

# Step 5: Analyzing Model Performance
from sklearn.metrics import accuracy_score

accuracy_rf = accuracy_score(y_test, y_pred_rf)
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)

print("Random Forest Accuracy:", accuracy_rf)
print("XGBoost Accuracy:", accuracy_xgb)
```



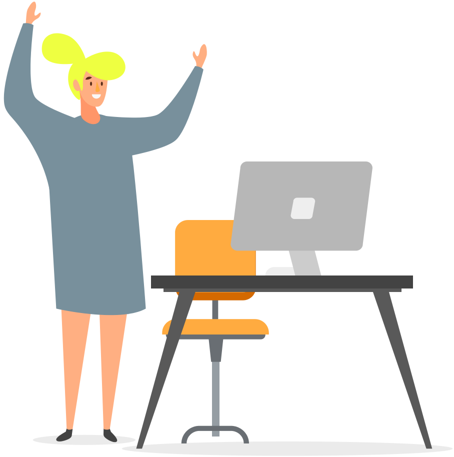
Text Preprocessing

The first step is to clean up the text and rectify the dataset by removing the irregularities in regular NLP Projects. As a result, we will do the following text-cleaning processes.

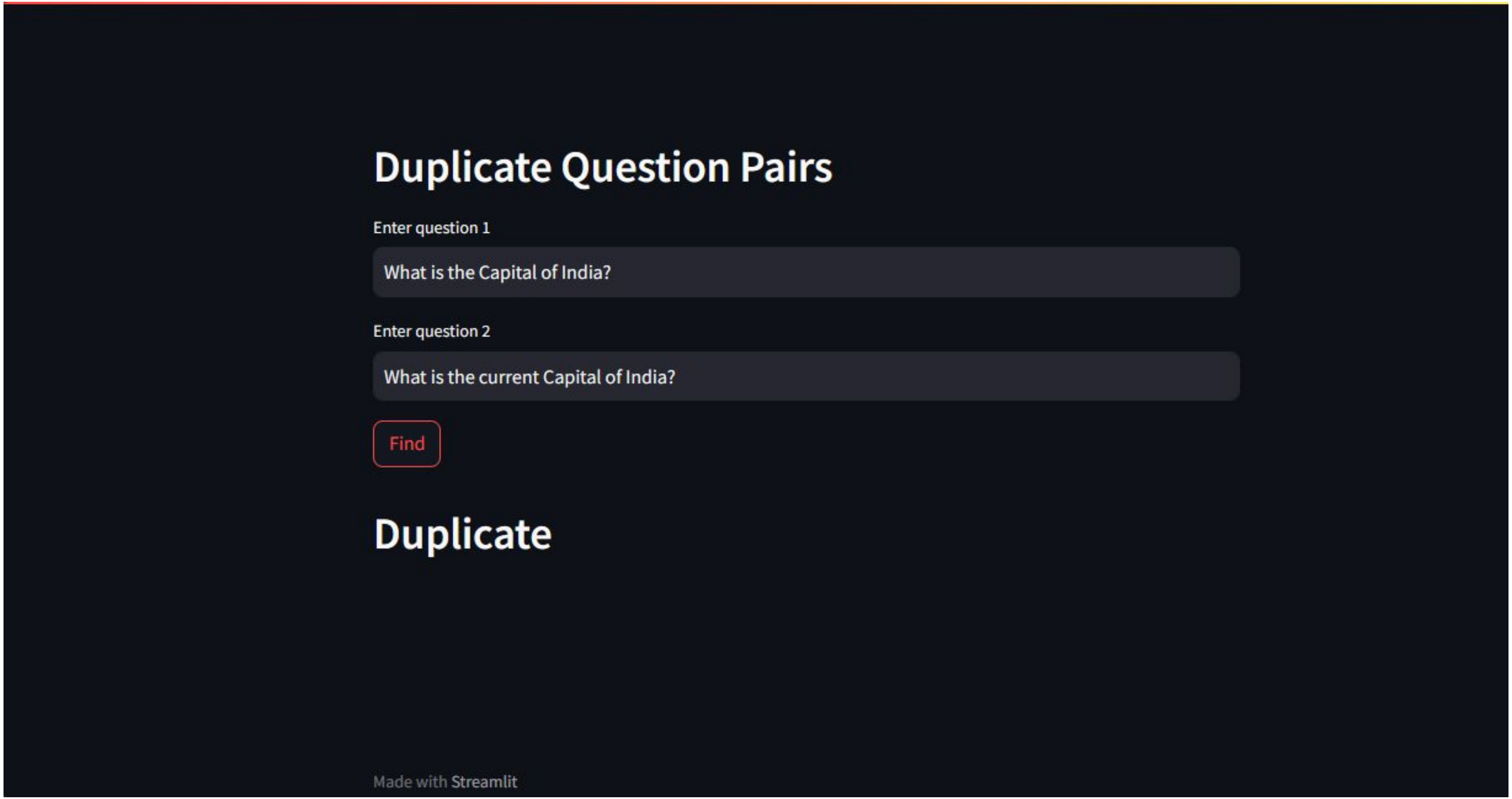
- **Lowercase:** If the text falls into one case, it is simple to vectorize and interpret because the vectorizer considers token and Token to be different words. So we will convert the entire text into lowercase.
- **String Equivalents:** The text contains multiple symbols, so we will replace them with corresponding string words.
- **Expand Contraction:** Contraction is written communication in human language to write words in short form. For example, don't stands for do not so there are multiple contractions which we need to change to corresponding complete forms.
- **Remove HTML tags:** The text contains some unnecessary HTML tags, so that we will remove them.
- **Remove Punctuation:** Punctuation is unnecessary and does not convey meaning, so it is better to remove them.

Selecting the Best Model

After preparing the data with 15 additional features, we trained both Random Forest and XGBoost models for an NLP project. Random Forest achieved approximately 78.7% accuracy, and XGBoost reached 79.2%. These optimizations improved performance by 2-2.5%. When the model falsely predicts non-duplicate values as duplicates, it can be a significant issue for user experience.



User Interface (Duplicate)



Duplicate Question Pairs

Enter question 1

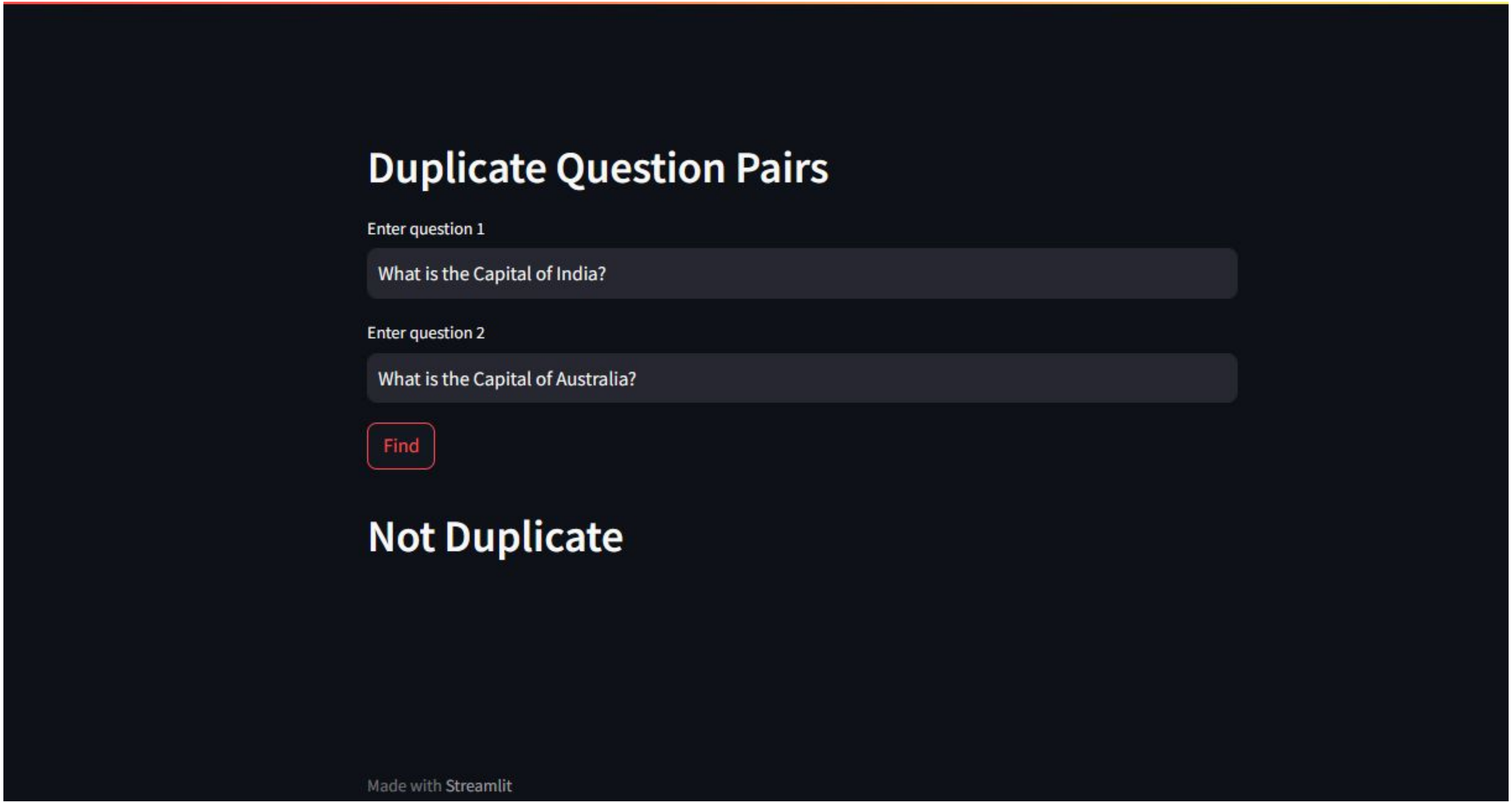
What is the Capital of India?

Enter question 2

What is the current Capital of India?

Duplicate

User Interface (Non-Duplicate)



Duplicate Question Pairs

Enter question 1

What is the Capital of India?

Enter question 2

What is the Capital of Australia?

Not Duplicate

THANKS