# Duplicate Question Pairs Detection Report

Tapas Mandal

## 1 Introduction

The objective of this project is to identify whether two questions are duplicates or not. The dataset is taken from Quora. The steps include preprocessing, feature engineering, visualization, dimensionality reduction, and classification using Random Forest and XGBoost.

## 2 Importing Libraries and Data

We start by importing necessary libraries and loading the dataset.

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('/content/questions.csv', on_bad_lines='skip')
df = df.sample(40000)
```

Here, we import Python libraries and load 40,000 random samples from the dataset for analysis.

## 3 Text Preprocessing

We clean text by expanding contractions, removing special characters, and handling HTML tags.

```python
from bs4 import BeautifulSoup
import re
```

```
def preprocess(q):
    q = str(q).lower().strip()
    q = q.replace('%', ' percent').replace('$', ' dollar'
       )
    q = BeautifulSoup(q, 'html.parser').get_text()
    q = re.sub(r'\W+', ' ', q).strip()
    return q

df['question1'] = df['question1'].apply(preprocess)
df['question2'] = df['question2'].apply(preprocess)
```

This ensures the text is cleaned and standardized before feature extraction.

## 4 Feature Engineering

We generate token-based, length-based, and fuzzy features to capture similarity between questions.

```
from nltk.corpus import stopwords
from fuzzywuzzy import fuzz
import distance
import nltk
nltk.download('stopwords')

# Example: Length features
def fetch_length_features(row):
    q1_tokens = row['question1'].split()
    q2_tokens = row['question2'].split()
    return [abs(len(q1_tokens)-len(q2_tokens)),
            (len(q1_tokens)+len(q2_tokens))/2]
```

Features like common words, fuzzy ratios, and substring lengths are extracted to represent question pairs numerically.

## 5 Dimensionality Reduction and Visualization

We apply t-SNE to reduce features to 2D/3D for visualization.

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.manifold import TSNE

X = MinMaxScaler().fit_transform(new_df[features])
tsne2d = TSNE(n_components=2).fit_transform(X)
```

t-SNE helps visualize how duplicate and non-duplicate questions cluster in feature space.

## 6 Model Training

We train Random Forest and XGBoost classifiers on the processed features.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from xgboost import XGBClassifier

x_train,x_test,y_train,y_test = train_test_split(
    final_df.iloc[:,1:], final_df.iloc[:,0], test_size
       =0.2, random_state=42)

rf = RandomForestClassifier()
rf.fit(x_train, y_train)
print("RF Accuracy:", accuracy_score(y_test, rf.predict(
   x_test)))

xgb = XGBClassifier()
xgb.fit(x_train, y_train)
print("XGB Accuracy:", accuracy_score(y_test, xgb.predict
   (x_test)))
```

Both models are evaluated. Random Forest and XGBoost provide strong performance for classification.

## 7 Conclusion

The project successfully applied preprocessing, feature engineering, visualization, and machine learning models to detect duplicate questions. The models achieved good accuracy, demonstrating the effectiveness of the engineered features.