```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import re
from bs4 import BeautifulSoup

import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('/content/drive/MyDrive/Dataset ML LAB 205122021/train.csv')
```

```
new_df = df.sample(30000,random_state=2)
```

```
new_df.head()
```

	id	qid1	qid2	question1	question2	is_duplicate
398782	398782	496695	532029	What is the best marketing automation tool for...	What is the best marketing automation tool for...	1
115086	115086	187729	187730	I am poor but I want to invest. What should I do?	I am quite poor and I want to be very rich. Wh...	0
327711	327711	454161	454162	I am from India and live abroad. I met a guy f...	T.I.E.T to Thapar University to Thapar	0

```
def preprocess(q):

    q = str(q).lower().strip()

    # Replace certain special characters with their string equivalents
    q = q.replace('%', ' percent')
    q = q.replace('$', ' dollar ')
    q = q.replace('₹', ' rupee ')
    q = q.replace('€', ' euro ')
    q = q.replace('@', ' at ')

    # The pattern '[math]' appears around 900 times in the whole dataset.
    q = q.replace('[math]', '')

    # Replacing some numbers with string equivalents (not perfect, can be done better to acc
    q = q.replace(',000,000,000 ', 'b ')
    q = q.replace(',000,000 ', 'm ')
    q = q.replace(',000 ', 'k ')
    q = re.sub(r'([0-9]+)000000000', r'\1b', q)
    q = re.sub(r'([0-9]+)000000', r'\1m', q)
    q = re.sub(r'([0-9]+)000', r'\1k', q)

    # Decontracting words
    # https://en.wikipedia.org/wiki/Wikipedia%3aList_of_English_contractions
    # https://stackoverflow.com/a/19794953
    contractions = {
    "ain't": "am not",
    "aren't": "are not",
    "can't": "can not",
    "can't've": "can not have",
    "'cause": "because",
    "could've": "could have",
    "couldn't": "could not",
    "couldn't've": "could not have",
    "didn't": "did not",
    "doesn't": "does not",
    "don't": "do not",
    "hadn't": "had not",
    "hadn't've": "had not have",
    "hasn't": "has not",
    "haven't": "have not",
    "he'd": "he would",
    "he'd've": "he would have",
    "he'll": "he will",
    "he'll've": "he will have",
    "he's": "he is",
    "how'd": "how did",
    "how'd'y": "how do you",
    "how'll": "how will",
    "how's": "how is",
    "i'd": "i would",
```

```
    "i'd've": "i would have",
    "i'll": "i will",
    "i'll've": "i will have",
    "i'm": "i am",
    "i've": "i have",
    "isn't": "is not",
    "it'd": "it would",
    "it'd've": "it would have",
    "it'll": "it will",
    "it'll've": "it will have",
    "it's": "it is",
    "let's": "let us",
    "ma'am": "madam",
    "mayn't": "may not",
    "might've": "might have",
    "mightn't": "might not",
    "mightn't've": "might not have",
    "must've": "must have",
    "mustn't": "must not",
    "mustn't've": "must not have",
    "needn't": "need not",
    "needn't've": "need not have",
    "o'clock": "of the clock",
    "oughtn't": "ought not",
    "oughtn't've": "ought not have",
    "shan't": "shall not",
    "sha'n't": "shall not",
    "shan't've": "shall not have",
    "she'd": "she would",
    "she'd've": "she would have",
    "she'll": "she will",
    "she'll've": "she will have",
    "she's": "she is",
    "should've": "should have",
    "shouldn't": "should not",
    "shouldn't've": "should not have",
    "so've": "so have",
    "so's": "so as",
    "that'd": "that would",
    "that'd've": "that would have",
    "that's": "that is",
    "there'd": "there would",
    "there'd've": "there would have",
    "there's": "there is",
    "they'd": "they would",
    "they'd've": "they would have",
    "they'll": "they will",
    "they'll've": "they will have",
    "they're": "they are",
    "they've": "they have",
    "to've": "to have",
```

```
    "wasn't": "was not",
    "we'd": "we would",
    "we'd've": "we would have",
    "we'll": "we will",
    "we'll've": "we will have",
    "we're": "we are",
    "we've": "we have",
    "weren't": "were not",
    "what'll": "what will",
    "what'll've": "what will have",
    "what're": "what are",
    "what's": "what is",
    "what've": "what have",
    "when's": "when is",
    "when've": "when have",
    "where'd": "where did",
    "where's": "where is",
    "where've": "where have",
    "who'll": "who will",
    "who'll've": "who will have",
    "who's": "who is",
    "who've": "who have",
    "why's": "why is",
    "why've": "why have",
    "will've": "will have",
    "won't": "will not",
    "won't've": "will not have",
    "would've": "would have",
    "wouldn't": "would not",
    "wouldn't've": "would not have",
    "y'all": "you all",
    "y'all'd": "you all would",
    "y'all'd've": "you all would have",
    "y'all're": "you all are",
    "y'all've": "you all have",
    "you'd": "you would",
    "you'd've": "you would have",
    "you'll": "you will",
    "you'll've": "you will have",
    "you're": "you are",
    "you've": "you have"
    }

    q_decontracted = []

    for word in q.split():
        if word in contractions:
            word = contractions[word]

        q_decontracted.append(word)
```

```
    q = ' '.join(q_decontracted)
    q = q.replace("'ve", " have")
    q = q.replace("n't", " not")
    q = q.replace("'re", " are")
    q = q.replace("'ll", " will")

    # Removing HTML tags
    q = BeautifulSoup(q)
    q = q.get_text()

    # Remove punctuations
    pattern = re.compile('\W')
    q = re.sub(pattern, ' ', q).strip()


    return q
```

```
preprocess("I've already! wasn't <b>done</b>?")
```

```
'i have already  was not done'
```

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

```
new_df.head()
```

	id	qid1	qid2	question1	question2	is_duplicate
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0
327711	327711	454161	454162	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar	0

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

```
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()
```

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1	75	76	
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0	48	56	
327711	327711	454161	454162	i am from india and live abroad i met a	t i e t to thapar university	0	104	119	
367788	367788	498109	491396	why do so many people in the u s hate	my boyfriend doesnt feel guilty	0	58	145	

```
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()
```

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1	75	76	
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0	48	56	
327711	327711	454161	454162	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar univers...	0	104	119	
367788	367788	498109	491396	why do so many people in the u s hate	my boyfriend doesnt feel guilty	0	58	145	

```
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))
```

151235	151235	237843	50930	of bhopal gas	behind the	0	34	49

```
new_df['word_total'] = new_df.apply(total_words, axis=1)
new_df.head()
```

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1	75	76	
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0	48	56	
327711	327711	454161	454162	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar univers...	0	104	119	
367788	367788	498109	491396	why do so many people in the u s hate the sou...	my boyfriend doesnt feel guilty when he hurts ...	0	58	145	
151235	151235	237843	50930	consequences of bhopal gas tragedy	what was the reason behind the bhopal gas tragedy	0	34	49	

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

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1	75	76	
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0	48	56	
327711	327711	454161	454162	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar univers...	0	104	119	
367788	367788	498109	491396	why do so many people in the u s hate the sou...	my boyfriend doesnt feel guilty when he hurts ...	0	58	145	
151235	151235	237843	50930	consequences of bhopal gas tragedy	what was the reason behind the bhopal gas tragedy	0	34	49	

```
!pip install nltk
import nltk
nltk.download('stopwords')
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nl
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from r
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nlt
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
from nltk.corpus import stopwords
```

```
# Advanced Features
from nltk.corpus import stopwords

def fetch_token_features(row):

    q1 = row['question1']
    q2 = row['question2']

    SAFE_DIV = 0.0001

    STOP_WORDS = stopwords.words("english")

    token_features = [0.0]*8

    # Converting the Sentence into Tokens:
    q1_tokens = q1.split()
    q2_tokens = q2.split()

    if len(q1_tokens) == 0 or len(q2_tokens) == 0:
        return token_features

    # Get the non-stopwords in Questions
    q1_words = set([word for word in q1_tokens if word not in STOP_WORDS])
    q2_words = set([word for word in q2_tokens if word not in STOP_WORDS])

    #Get the stopwords in Questions
    q1_stops = set([word for word in q1_tokens if word in STOP_WORDS])
    q2_stops = set([word for word in q2_tokens if word in STOP_WORDS])

    # Get the common non-stopwords from Question pair
    common_word_count = len(q1_words.intersection(q2_words))

    # Get the common stopwords from Question pair
    common_stop_count = len(q1_stops.intersection(q2_stops))

    # Get the common Tokens from Question pair
    common_token_count = len(set(q1_tokens).intersection(set(q2_tokens)))


    token_features[0] = common_word_count / (min(len(q1_words), len(q2_words)) + SAFE_DIV)
    token_features[1] = common_word_count / (max(len(q1_words), len(q2_words)) + SAFE_DIV)
    token_features[2] = common_stop_count / (min(len(q1_stops), len(q2_stops)) + SAFE_DIV)
    token_features[3] = common_stop_count / (max(len(q1_stops), len(q2_stops)) + SAFE_DIV)
    token_features[4] = common_token_count / (min(len(q1_tokens), len(q2_tokens)) + SAFE_DIV
    token_features[5] = common_token_count / (max(len(q1_tokens), len(q2_tokens)) + SAFE_DIV

    # Last word of both question is same or not
    token_features[6] = int(q1_tokens[-1] == q2_tokens[-1])

    # First word of both question is same or not
    token_features[7] = int(q1_tokens[0] == q2_tokens[0])
```

```
    return token_features
```

Start coding or generate with AI.

```
token_features = new_df.apply(fetch_token_features, axis=1)

new_df["cwc_min"]       = list(map(lambda x: x[0], token_features))
new_df["cwc_max"]       = list(map(lambda x: x[1], token_features))
new_df["csc_min"]       = list(map(lambda x: x[2], token_features))
new_df["csc_max"]       = list(map(lambda x: x[3], token_features))
new_df["ctc_min"]       = list(map(lambda x: x[4], token_features))
new_df["ctc_max"]       = list(map(lambda x: x[5], token_features))
new_df["last_word_eq"]  = list(map(lambda x: x[6], token_features))
new_df["first_word_eq"] = list(map(lambda x: x[7], token_features))
```

```
new_df.head()
```

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1	75	76	
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0	48	56	
327711	327711	454161	454162	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar univers...	0	104	119	
367788	367788	498109	491396	why do so many people in the u s hate the sou...	my boyfriend doesnt feel guilty when he hurts ...	0	58	145	
151235	151235	237843	50930	consequences of bhopal gas tragedy	what was the reason behind the bhopal gas tragedy	0	34	49	

5 rows × 21 columns

```
!pip install distance
```

```
Collecting distance
  Downloading Distance-0.1.3.tar.gz (180 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 180.3/180.3 kB 2.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: distance
  Building wheel for distance (setup.py) ... done
  Created wheel for distance: filename=Distance-0.1.3-py3-none-any.whl size=16258 sha256
  Stored in directory: /root/.cache/pip/wheels/e8/bb/de/f71bf63559ea9a921059a5405806f7ff
Successfully built distance
Installing collected packages: distance
Successfully installed distance-0.1.3
```

```
import distance

def fetch_length_features(row):

    q1 = row['question1']
    q2 = row['question2']

    length_features = [0.0]*3

    # Converting the Sentence into Tokens:
    q1_tokens = q1.split()
    q2_tokens = q2.split()

    if len(q1_tokens) == 0 or len(q2_tokens) == 0:
        return length_features

    # Absolute length features
    length_features[0] = abs(len(q1_tokens) - len(q2_tokens))

    #Average Token Length of both Questions
    length_features[1] = (len(q1_tokens) + len(q2_tokens))/2

    strs = list(distance.lcsubstrings(q1, q2))
    length_features[2] = len(strs[0]) / (min(len(q1), len(q2)) + 1)

    return length_features



length_features = new_df.apply(fetch_length_features, axis=1)

new_df['abs_len_diff'] = list(map(lambda x: x[0], length_features))
new_df['mean_len'] = list(map(lambda x: x[1], length_features))
new_df['longest_substr_ratio'] = list(map(lambda x: x[2], length_features))


new_df.head()
```

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1	75	76	
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0	48	56	
327711	327711	454161	454162	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar univers...	0	104	119	
367788	367788	498109	491396	why do so many people in the u s hate the sou...	my boyfriend doesnt feel guilty when he hurts ...	0	58	145	
151235	151235	237843	50930	consequences of bhopal gas tragedy	what was the reason behind the bhopal gas tragedy	0	34	49	

5 rows × 24 columns

```
!pip install fuzzywuzzy
```

```
Collecting fuzzywuzzy
  Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl (18 kB)
Installing collected packages: fuzzywuzzy
Successfully installed fuzzywuzzy-0.18.0
```

```
# Fuzzy Features
from fuzzywuzzy import fuzz

def fetch_fuzzy_features(row):

    q1 = row['question1']
    q2 = row['question2']

    fuzzy_features = [0.0]*4

    # fuzz_ratio
    fuzzy_features[0] = fuzz.QRatio(q1, q2)

    # fuzz_partial_ratio
    fuzzy_features[1] = fuzz.partial_ratio(q1, q2)

    # token_sort_ratio
    fuzzy_features[2] = fuzz.token_sort_ratio(q1, q2)

    # token_set_ratio
    fuzzy_features[3] = fuzz.token_set_ratio(q1, q2)

    return fuzzy_features


fuzzy_features = new_df.apply(fetch_fuzzy_features, axis=1)

# Creating new feature columns for fuzzy features
new_df['fuzz_ratio'] = list(map(lambda x: x[0], fuzzy_features))
new_df['fuzz_partial_ratio'] = list(map(lambda x: x[1], fuzzy_features))
new_df['token_sort_ratio'] = list(map(lambda x: x[2], fuzzy_features))
new_df['token_set_ratio'] = list(map(lambda x: x[3], fuzzy_features))


print(new_df.shape)
new_df.head()
```

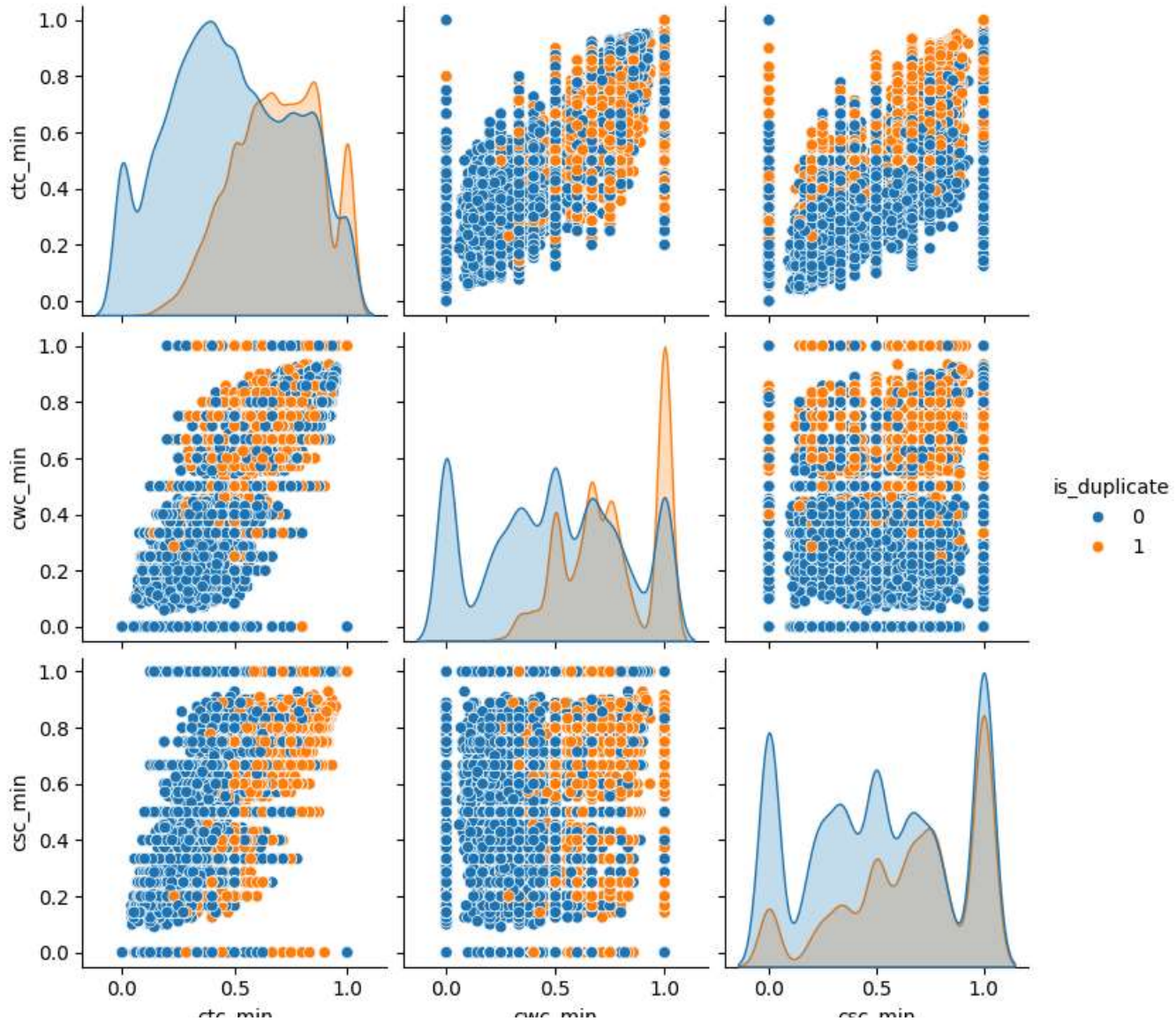
```
(30000, 28)
```

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1	75	76	
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0	48	56	
327711	327711	454161	454162	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar univers...	0	104	119	
367788	367788	498109	491396	why do so many people in the u s hate the sou...	my boyfriend doesnt feel guilty when he hurts ...	0	58	145	
151235	151235	237843	50930	consequences of bhopal gas tragedy	what was the reason behind the bhopal gas tragedy	0	34	49	

5 rows × 28 columns

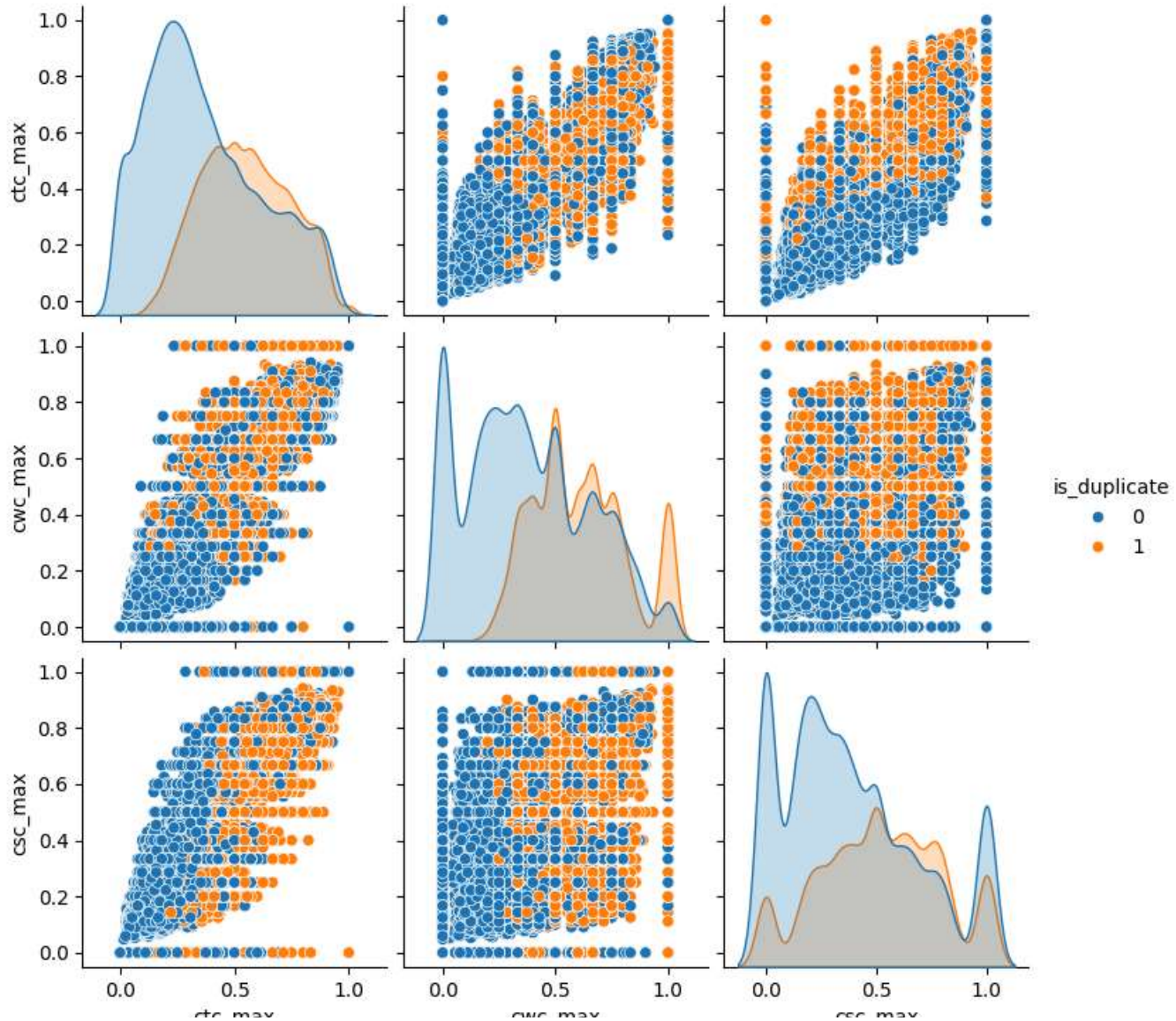
```
sns.pairplot(new_df[['ctc_min', 'cwc_min', 'csc_min', 'is_duplicate']],hue='is_duplicate')
```

```
<seaborn.axisgrid.PairGrid at 0x7d535e5f69e0>
```



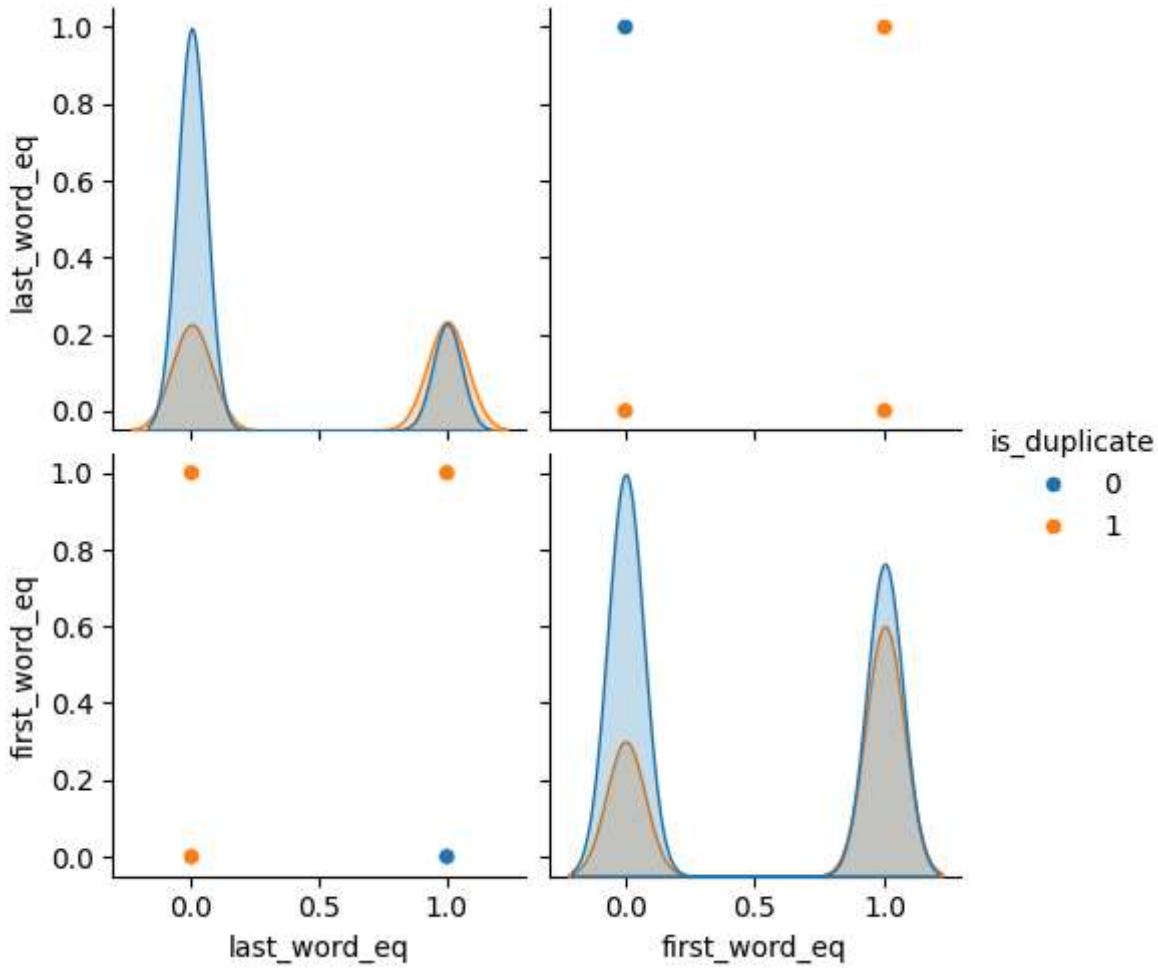

```
sns.pairplot(new_df[['ctc_max', 'cwc_max', 'csc_max', 'is_duplicate']],hue='is_duplicate')
```

```
<seaborn.axisgrid.PairGrid at 0x7d535a395480>
```



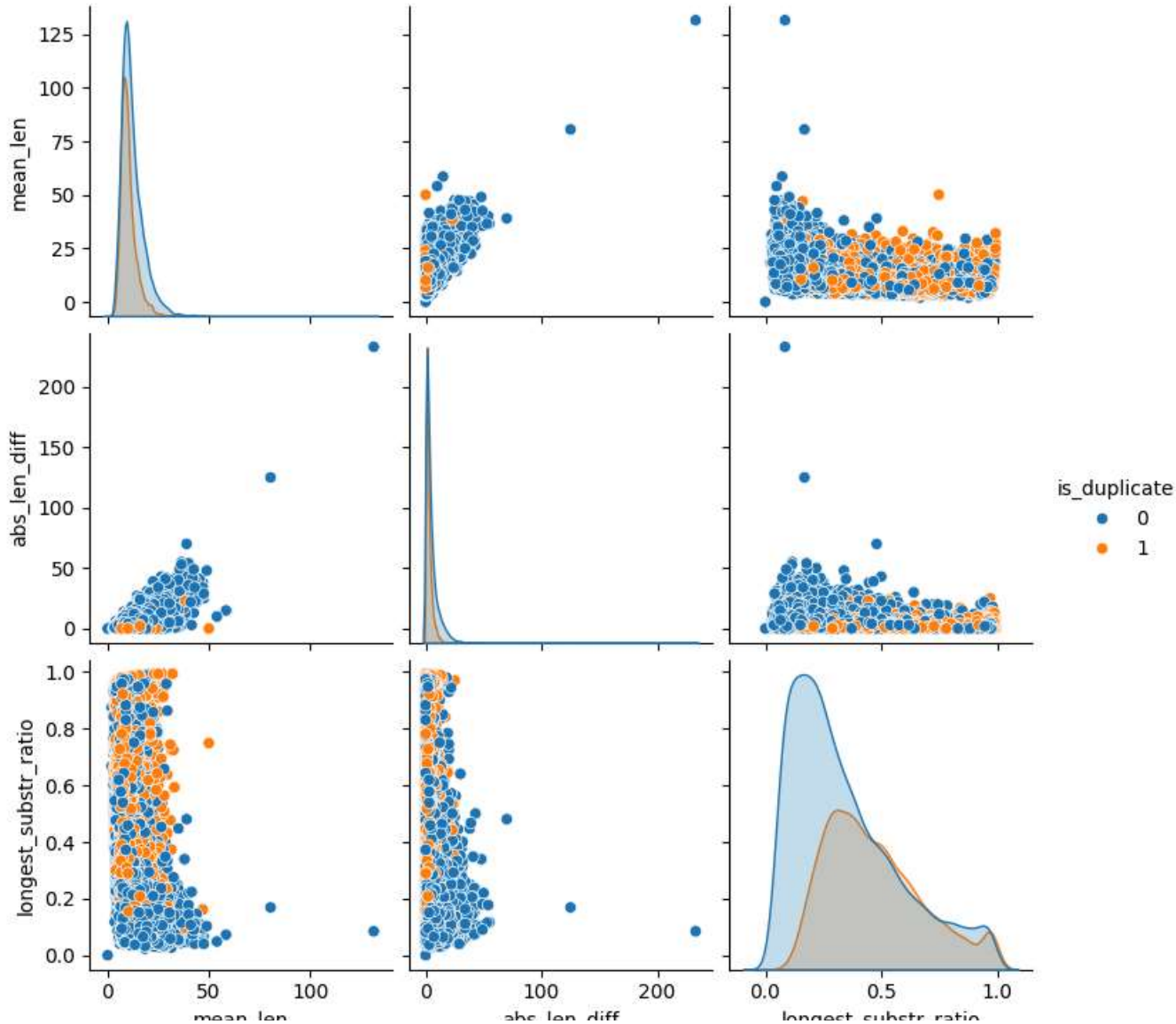

```
sns.pairplot(new_df[['last_word_eq', 'first_word_eq', 'is_duplicate']],hue='is_duplicate')
```

```
<seaborn.axisgrid.PairGrid at 0x7d5358cb69e0>
```



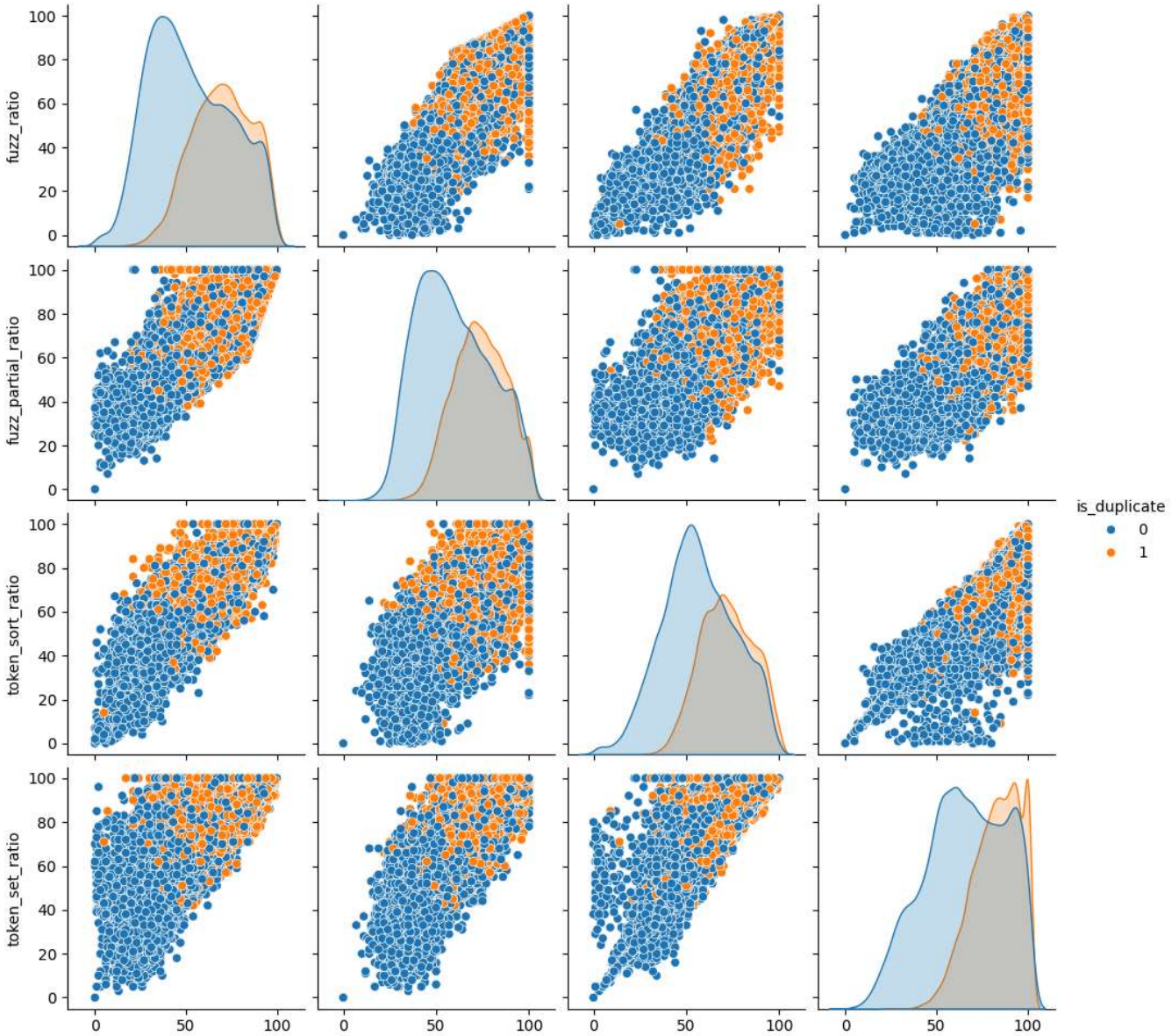

```
sns.pairplot(new_df[['mean_len', 'abs_len_diff','longest_substr_ratio', 'is_duplicate']],hue
```

```
<seaborn.axisgrid.PairGrid at 0x7d5359525330>
```




```
sns.pairplot(new_df[['fuzz_ratio', 'fuzz_partial_ratio','token_sort_ratio','token_set_ratio'
```

```
<seaborn.axisgrid.PairGrid at 0x7d5352c2f520>
```



```
# Using TSNE for Dimentionality reduction for 15 Features(Generated after cleaning the data)

from sklearn.preprocessing import MinMaxScaler

X = MinMaxScaler().fit_transform(new_df[['cwc_min', 'cwc_max', 'csc_min', 'csc_max' , 'ctc_m
y = new_df['is_duplicate'].values


from sklearn.manifold import TSNE

tsne2d = TSNE(
    n_components=2,
    init='random', # pca
    random_state=101,
    method='barnes_hut',
    n_iter=1000,
    verbose=2,
    angle=0.5
).fit_transform(X)
```

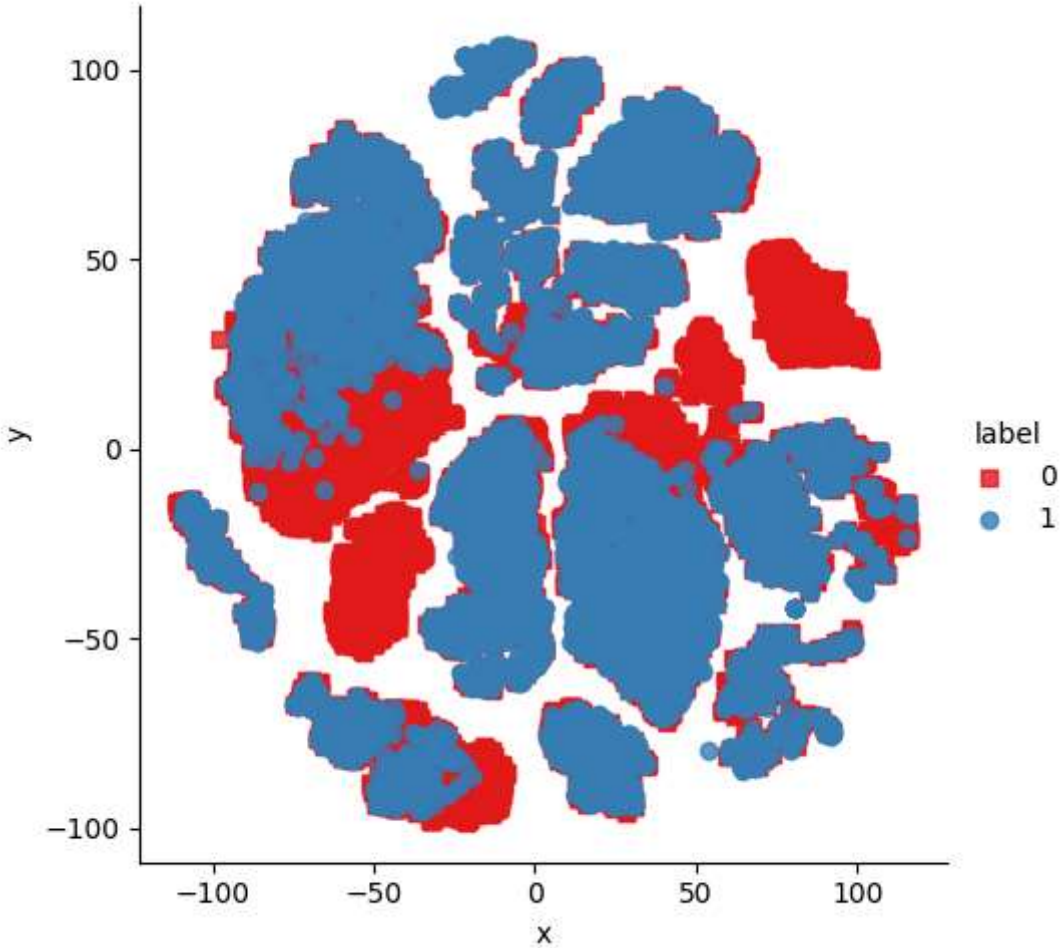
```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 30000 samples in 0.084s...
[t-SNE] Computed neighbors for 30000 samples in 8.403s...
[t-SNE] Computed conditional probabilities for sample 1000 / 30000
[t-SNE] Computed conditional probabilities for sample 2000 / 30000
[t-SNE] Computed conditional probabilities for sample 3000 / 30000
[t-SNE] Computed conditional probabilities for sample 4000 / 30000
[t-SNE] Computed conditional probabilities for sample 5000 / 30000
[t-SNE] Computed conditional probabilities for sample 6000 / 30000
[t-SNE] Computed conditional probabilities for sample 7000 / 30000
[t-SNE] Computed conditional probabilities for sample 8000 / 30000
[t-SNE] Computed conditional probabilities for sample 9000 / 30000
[t-SNE] Computed conditional probabilities for sample 10000 / 30000
[t-SNE] Computed conditional probabilities for sample 11000 / 30000
[t-SNE] Computed conditional probabilities for sample 12000 / 30000
[t-SNE] Computed conditional probabilities for sample 13000 / 30000
[t-SNE] Computed conditional probabilities for sample 14000 / 30000
[t-SNE] Computed conditional probabilities for sample 15000 / 30000
[t-SNE] Computed conditional probabilities for sample 16000 / 30000
[t-SNE] Computed conditional probabilities for sample 17000 / 30000
[t-SNE] Computed conditional probabilities for sample 18000 / 30000
[t-SNE] Computed conditional probabilities for sample 19000 / 30000
[t-SNE] Computed conditional probabilities for sample 20000 / 30000
[t-SNE] Computed conditional probabilities for sample 21000 / 30000
[t-SNE] Computed conditional probabilities for sample 22000 / 30000
[t-SNE] Computed conditional probabilities for sample 23000 / 30000
[t-SNE] Computed conditional probabilities for sample 24000 / 30000
[t-SNE] Computed conditional probabilities for sample 25000 / 30000
[t-SNE] Computed conditional probabilities for sample 26000 / 30000
[t-SNE] Computed conditional probabilities for sample 27000 / 30000
[t-SNE] Computed conditional probabilities for sample 28000 / 30000
[t-SNE] Computed conditional probabilities for sample 29000 / 30000
```

```
[t-SNE] Computed conditional probabilities for sample 30000 / 30000
[t-SNE] Mean sigma: 0.080413
[t-SNE] Computed conditional probabilities in 1.072s
[t-SNE] Iteration 50: error = 109.7805786, gradient norm = 0.0225089 (50 iterations i
[t-SNE] Iteration 100: error = 89.5831604, gradient norm = 0.0074239 (50 iterations i
[t-SNE] Iteration 150: error = 85.6243744, gradient norm = 0.0047372 (50 iterations i
[t-SNE] Iteration 200: error = 83.7841568, gradient norm = 0.0038870 (50 iterations i
[t-SNE] Iteration 250: error = 82.6397934, gradient norm = 0.0029084 (50 iterations i
[t-SNE] KL divergence after 250 iterations with early exaggeration: 82.639793
[t-SNE] Iteration 300: error = 3.3938556, gradient norm = 0.0073763 (50 iterations in
[t-SNE] Iteration 350: error = 2.7820463, gradient norm = 0.0073553 (50 iterations in
[t-SNE] Iteration 400: error = 2.4498415, gradient norm = 0.0068701 (50 iterations in
[t-SNE] Iteration 450: error = 2.2408490, gradient norm = 0.0064593 (50 iterations in
[t-SNE] Iteration 500: error = 2.0968621, gradient norm = 0.0061067 (50 iterations in
[t-SNE] Iteration 550: error = 1.9919664, gradient norm = 0.0057804 (50 iterations in
[t-SNE] Iteration 600: error = 1.9128428, gradient norm = 0.0053741 (50 iterations in
[t-SNE] Iteration 650: error = 1.8518257, gradient norm = 0.0050063 (50 iterations in
[t-SNE] Iteration 700: error = 1.8034400, gradient norm = 0.0046630 (50 iterations in
[t-SNE] Iteration 750: error = 1.7641261, gradient norm = 0.0043631 (50 iterations in
[t-SNE] Iteration 800: error = 1.7315955, gradient norm = 0.0040643 (50 iterations in
[t-SNE] Iteration 850: error = 1.7043265, gradient norm = 0.0038040 (50 iterations in
[t-SNE] Iteration 900: error = 1.6811551, gradient norm = 0.0035603 (50 iterations in
[t-SNE] Iteration 950: error = 1.6613963, gradient norm = 0.0033345 (50 iterations in
[t-SNE] Iteration 1000: error = 1.6442907, gradient norm = 0.0031099 (50 iterations i
```

```
x_df = pd.DataFrame({'x':tsne2d[:,0], 'y':tsne2d[:,1] ,'label':y})

# draw the plot in appropriate place in the grid
sns.lmplot(data=x_df, x='x', y='y', hue='label', fit_reg=False,palette="Set1",markers=['s','
```

```
<seaborn.axisgrid.FacetGrid at 0x7d5352266770>
```




```
tsne3d = TSNE(
    n_components=3,
    init='random', # pca
    random_state=101,
    method='barnes_hut',
    n_iter=1000,
    verbose=2,
    angle=0.5
).fit_transform(X)
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 30000 samples in 0.132s...
[t-SNE] Computed neighbors for 30000 samples in 6.822s...
[t-SNE] Computed conditional probabilities for sample 1000 / 30000
[t-SNE] Computed conditional probabilities for sample 2000 / 30000
[t-SNE] Computed conditional probabilities for sample 3000 / 30000
[t-SNE] Computed conditional probabilities for sample 4000 / 30000
[t-SNE] Computed conditional probabilities for sample 5000 / 30000
[t-SNE] Computed conditional probabilities for sample 6000 / 30000
[t-SNE] Computed conditional probabilities for sample 7000 / 30000
[t-SNE] Computed conditional probabilities for sample 8000 / 30000
[t-SNE] Computed conditional probabilities for sample 9000 / 30000
[t-SNE] Computed conditional probabilities for sample 10000 / 30000
[t-SNE] Computed conditional probabilities for sample 11000 / 30000
```

```
[t-SNE] Computed conditional probabilities for sample 12000 / 30000
[t-SNE] Computed conditional probabilities for sample 13000 / 30000
[t-SNE] Computed conditional probabilities for sample 14000 / 30000
[t-SNE] Computed conditional probabilities for sample 15000 / 30000
[t-SNE] Computed conditional probabilities for sample 16000 / 30000
[t-SNE] Computed conditional probabilities for sample 17000 / 30000
[t-SNE] Computed conditional probabilities for sample 18000 / 30000
[t-SNE] Computed conditional probabilities for sample 19000 / 30000
[t-SNE] Computed conditional probabilities for sample 20000 / 30000
[t-SNE] Computed conditional probabilities for sample 21000 / 30000
[t-SNE] Computed conditional probabilities for sample 22000 / 30000
[t-SNE] Computed conditional probabilities for sample 23000 / 30000
[t-SNE] Computed conditional probabilities for sample 24000 / 30000
[t-SNE] Computed conditional probabilities for sample 25000 / 30000
[t-SNE] Computed conditional probabilities for sample 26000 / 30000
[t-SNE] Computed conditional probabilities for sample 27000 / 30000
[t-SNE] Computed conditional probabilities for sample 28000 / 30000
[t-SNE] Computed conditional probabilities for sample 29000 / 30000
[t-SNE] Computed conditional probabilities for sample 30000 / 30000
[t-SNE] Mean sigma: 0.080413
[t-SNE] Computed conditional probabilities in 1.043s
[t-SNE] Iteration 50: error = 110.3409729, gradient norm = 0.0149619 (50 iterations i
[t-SNE] Iteration 100: error = 86.4499359, gradient norm = 0.0045841 (50 iterations i
[t-SNE] Iteration 150: error = 83.2826080, gradient norm = 0.0022951 (50 iterations i
[t-SNE] Iteration 200: error = 82.0343475, gradient norm = 0.0015075 (50 iterations i
[t-SNE] Iteration 250: error = 81.3193817, gradient norm = 0.0011174 (50 iterations i
[t-SNE] KL divergence after 250 iterations with early exaggeration: 81.319382
[t-SNE] Iteration 300: error = 2.9794178, gradient norm = 0.0038859 (50 iterations in
[t-SNE] Iteration 350: error = 2.3706670, gradient norm = 0.0030978 (50 iterations in
[t-SNE] Iteration 400: error = 2.0590405, gradient norm = 0.0024685 (50 iterations in
[t-SNE] Iteration 450: error = 1.8746790, gradient norm = 0.0021109 (50 iterations in
[t-SNE] Iteration 500: error = 1.7539742, gradient norm = 0.0017413 (50 iterations in
[t-SNE] Iteration 550: error = 1.6689844, gradient norm = 0.0015294 (50 iterations in
[t-SNE] Iteration 600: error = 1.6069720, gradient norm = 0.0013156 (50 iterations in
[t-SNE] Iteration 650: error = 1.5599891, gradient norm = 0.0011492 (50 iterations in
[t-SNE] Iteration 700: error = 1.5253487, gradient norm = 0.0009487 (50 iterations in
[t-SNE] Iteration 750: error = 1.5004241, gradient norm = 0.0007620 (50 iterations in
[t-SNE] Iteration 800: error = 1.4822491, gradient norm = 0.0006269 (50 iterations in
[t-SNE] Iteration 850: error = 1.4683144, gradient norm = 0.0005383 (50 iterations in
[t-SNE] Iteration 900: error = 1.4571249, gradient norm = 0.0004518 (50 iterations in
[t-SNE] Iteration 950: error = 1.4490130, gradient norm = 0.0003456 (50 iterations in
[t-SNE] Iteration 1000: error = 1.4427356, gradient norm = 0.0003044 (50 iterations i
[t-SNE] KL divergence after 1000 iterations: 1.442736
```

```
import plotly.graph_objs as go
import plotly.tools as tls
import plotly.offline as py
py.init_notebook_mode(connected=True)

trace1 = go.Scatter3d(
    x=tsne3d[:,0],
    y=tsne3d[:,1],
    z=tsne3d[:,2],
    mode='markers',
```

```
        marker=dict(
            sizemode='diameter',
            color = y,
            colorscale = 'Portland',
            colorbar = dict(title = 'duplicate'),
            line=dict(color='rgb(255, 255, 255)'),
            opacity=0.75
        )
    )

    data=[trace1]
    layout=dict(height=800, width=800, title='3d embedding with engineered features')
    fig=dict(data=data, layout=layout)
    py.iplot(fig, filename='3DBubble')
```