Experiment 6 and 8: One Hot Encoder, TF-IDF

#Objective:

To explore different encoding techniques for text data, specifically One-Hot Encoding and TF-IDF (Term Frequency-Inverse Document Frequency), and analyze the vectorized representation of words using pre-trained word embeddings from Google News.

#Code

```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder()
one_hot_encoded = ohe.fit_transform(df[["text"]])
print(one_hot_encoded)
col = ohe.get_feature_names_out(['text'])
print(col)
cat = ohe.categories_
print(cat)
```

```
<Compressed Sparse Row sparse matrix of dtype 'float64'
	with 4 stored elements and shape (4, 4)>
  Coords	Values
  (0, 2)	1.0
  (1, 0)	1.0
  (2, 3)	1.0
  (3, 1)	1.0
['text_Facebook watch facebook' 'text_Facebook write comments'
 'text_People Watch Facebook' 'text_People write comments']
[array(['Facebook watch facebook', 'Facebook write comments',
       'People Watch Facebook', 'People write comments'], dtype=object)]
```

```
arr = one_hot_encoded.toarray()
df1 = pd.DataFrame(arr, columns=cat)
df2 = pd.concat([df, df1], axis=1)
df2
```

	text	output	(Facebook watch facebook,)	(Facebook write comments,)	(People Watch Facebook,)	(People write comments,)
0	People Watch Facebook	1	0.0	0.0	1.0	0.0
1	Facebook watch facebook	1	1.0	0.0	0.0	0.0
2	People write comments	0	0.0	0.0	0.0	1.0
3	Facebook write comments	0	0.0	1.0	0.0	0.0

```
!pip install gensim
import gensim
from gensim import models
import gensim.downloader as api
wv = api.load('word2vec-google-news-300')
vec_cricket = wv['cricket'] #it will have 300 values as 300 dimensions
vec_cricket
```

```
array([-3.67187500e-01, -1.21582031e-01,  2.85156250e-01,  8.15429688e-02,
```

3.19824219e-02, -3.19824219e-02, 1.34765625e-01, -2.73437500e-01,
9.46044922e-03, -1.07421875e-01, 2.48046875e-01, -6.05468750e-01,
5.02929688e-02, 2.98828125e-01, 9.57031250e-02, 1.39648438e-01,
-5.41992188e-02, 2.91015625e-01, 2.85156250e-01, 1.51367188e-01,
-2.89062500e-01, -3.46679688e-02, 1.81884766e-02, -3.92578125e-01,
2.46093750e-01, 2.51953125e-01, -9.86328125e-02, 3.22265625e-01,
4.49218750e-01, -1.36718750e-01, -2.34375000e-01, 4.12597656e-02,
-2.15820312e-01, 1.69921875e-01, 2.56347656e-02, 1.50146484e-02,
-3.75976562e-02, 6.95800781e-03, 4.00390625e-01, 2.09960938e-01,
1.17675781e-01, -4.19921875e-02, 2.34375000e-01, 2.03125000e-01,
-1.86523438e-01, -2.46093750e-01, 3.12500000e-01, -2.59765625e-01,
-1.06933594e-01, 1.04003906e-01, -1.79687500e-01, 5.71289062e-02,
-7.41577148e-03, -5.59082031e-02, 7.61718750e-02, -4.14062500e-01,
-3.65234375e-01, -3.35937500e-01, -1.54296875e-01, -2.39257812e-01,
-3.73046875e-01, 2.27355957e-03, -3.51562500e-01, 8.64257812e-02,
1.26953125e-01, 2.21679688e-01, -9.86328125e-02, 1.08886719e-01,
3.65234375e-01, -5.66406250e-02, 5.66406250e-02, -1.09375000e-01,
-1.66992188e-01, -4.54101562e-02, -2.00195312e-01, -1.22558594e-01,
1.31835938e-01, -1.31835938e-01, 1.03027344e-01, -3.41796875e-01,
-1.57226562e-01, 2.04101562e-01, 4.39453125e-02, 2.44140625e-01,
-3.19824219e-02, 3.20312500e-01, -4.41894531e-02, 1.08398438e-01,
-4.98046875e-02, -9.52148438e-03, 2.46093750e-01, -5.59082031e-02,
4.07714844e-02, -1.78222656e-02, -2.95410156e-02, 1.65039062e-01,
5.03906250e-01, -2.81250000e-01, 9.81445312e-02, 1.80664062e-02,
-1.83593750e-01, 2.53906250e-01, 2.25585938e-01, 1.63574219e-02,
1.81640625e-01, 1.38671875e-01, 3.33984375e-01, 1.39648438e-01,
1.45874023e-02, -2.89306641e-02, -8.39843750e-02, 1.50390625e-01,
1.67968750e-01, 2.28515625e-01, 3.59375000e-01, 1.22558594e-01,
-3.28125000e-01, -1.56250000e-01, 2.77343750e-01, 1.77001953e-02,
-1.46484375e-01, -4.51660156e-03, -4.46777344e-02, 1.75781250e-01,
-3.75000000e-01, 1.16699219e-01, -1.39648438e-01, 2.55859375e-01,
-1.96289062e-01, -2.57568359e-02, -5.41992188e-02, -2.51464844e-02,
-1.93359375e-01, -3.17382812e-02, -8.74023438e-02, -1.32812500e-01,
-2.12402344e-02, 4.33593750e-01, -5.20019531e-02, 3.46679688e-02,
8.00781250e-02, 3.41796875e-02, 1.99218750e-01, -2.39257812e-02,
-2.37304688e-01, 1.93359375e-01, 7.32421875e-02, -2.87109375e-01,
1.25000000e-01, 8.44726562e-02, 1.30859375e-01, -2.19726562e-01,
-1.61132812e-01, -2.63671875e-01, -5.46875000e-01, -2.96875000e-01,
3.44238281e-02, -2.87109375e-01, -1.93359375e-01, -1.61132812e-01,
-3.84765625e-01, -2.14843750e-01, -6.22558594e-03, -1.27929688e-01,
-1.00097656e-01, -6.21093750e-01, 3.78906250e-01, -4.58984375e-01,
1.44531250e-01, -9.13085938e-02, -3.08593750e-01, 2.23632812e-01,
7.86132812e-02, -2.16796875e-01, 8.78906250e-02, -1.66992188e-01,
1.14746094e-02, -2.53906250e-01, -6.25000000e-02, 6.04248047e-03,
1.56250000e-01, 4.37500000e-01, -2.23632812e-01, -2.32421875e-01,
2.75390625e-01, 2.39257812e-01, 4.49218750e-02, -7.51953125e-02,
5.74218750e-01, -2.61230469e-02, -1.21582031e-01, 2.44140625e-01,
-3.37890625e-01, 8.59375000e-02, -7.71484375e-02, 4.85839844e-02,

```
        1.43554688e-01,  4.25781250e-01, -4.29687500e-02, -1.08398438e-01,
        1.19628906e-01, -1.91406250e-01, -2.12890625e-01, -2.87109375e-01,
       -1.14746094e-01, -2.04101562e-01, -2.06298828e-02, -2.53906250e-01,
        8.25195312e-02, -3.97949219e-02, -1.57226562e-01,  1.34765625e-01,
        2.08007812e-01, -1.78710938e-01, -2.00195312e-02, -8.34960938e-02,
       -1.20605469e-01,  4.29687500e-02, -1.94335938e-01, -1.32812500e-01,
       -2.17285156e-02, -2.35351562e-01, -3.63281250e-01,  1.51367188e-01,
        9.32617188e-02,  1.63085938e-01,  1.02050781e-01, -4.27734375e-01,
        2.83203125e-01,  2.74658203e-04, -3.20312500e-01,  1.68457031e-02,
        4.06250000e-01, -5.24902344e-02,  7.91015625e-02, -1.41601562e-01,
        5.27343750e-01, -1.26953125e-01,  4.74609375e-01, -6.64062500e-02,
        3.41796875e-01, -1.78710938e-01,  3.69140625e-01, -2.05078125e-01,
        5.82885742e-03, -1.84570312e-01, -8.88671875e-02, -1.81640625e-01,
       -4.80957031e-02,  4.39453125e-01,  2.12890625e-01, -3.07617188e-02,
        9.32617188e-02,  2.40234375e-01,  2.39257812e-01,  2.51953125e-01,
       -1.98974609e-02,  1.24511719e-01, -4.73632812e-02, -2.13623047e-02,
        3.12500000e-02,  3.05175781e-02,  2.79296875e-01,  9.08203125e-02,
       -2.02148438e-01, -2.19726562e-02, -2.63671875e-01,  8.78906250e-02,
       -1.07421875e-01, -2.49023438e-01, -1.22070312e-02,  1.73828125e-01,
       -9.91210938e-02,  7.27539062e-02,  2.59765625e-01, -4.60937500e-01,
        3.59375000e-01, -2.25585938e-01,  1.87988281e-02, -2.19726562e-01,
       -2.08984375e-01, -1.51367188e-01,  8.64257812e-02,  1.11694336e-02,
        6.93359375e-02, -2.99072266e-02,  1.43554688e-01,  1.89453125e-01,
       -1.32812500e-01,  4.72656250e-01, -1.40625000e-01, -2.52685547e-02,
        1.91406250e-01, -2.63671875e-01, -1.39648438e-01,  1.09375000e-01,
        1.97753906e-02,  2.49023438e-01, -1.42578125e-01,  4.15039062e-02],
      dtype=float32)
```

```
vec_cricket.shape
(300,)
```

```
wv.most_similar('Google')
[('Google_Nasdaq_GOOG', 0.7819362878799438),
 ('Google_GOOG', 0.7756521105766296),
 ('Google_NASDAQ_GOOG', 0.7557772994041443),
 ('Google_NSDQ_GOOG', 0.7538513541221619),
 ('Yahoo', 0.7491979598999023),
 ('GoogleGoogle', 0.7281472682952881),
 ('search_engine', 0.7255110144615173),
 ('Google_nasdaq_GOOG', 0.7014852166175842),
 ('Baidu', 0.6993466019630432),
 ('NASDAQ_GOOG', 0.6812566518783569)]
```

```
wv.most_similar('AI')
[('Steven_Spielberg_Artificial_Intelligence', 0.5575934052467346),
 ('Index_MDE_##/###/####', 0.5415324568748474),
 ('Enemy_AI', 0.5256390571594238),
 ('Ace_Combat_Zero', 0.522663414478302),
```

```
('DOA4', 0.5182536840438843),
('mechs', 0.5137375593185425),
('mech', 0.5077533721923828),
('playstyle', 0.5072520971298218),
('AI_bots', 0.5051203370094299),
('deathmatch_mode', 0.504591703414917)]

wv.most_similar('Artificial_Intelligence')
[('artificial_intelligence', 0.7004302144050598),
 ('USA_Subjex_Corporation', 0.5363693833351135),
 ('Artificial_Intelligence_AI', 0.52828049659729),
 ('Novamente', 0.52488112449646),
 ('Computational_Intelligence', 0.5138623118400574),
 ('Ben_Goertzel', 0.5136528611183167),
 ('Bot_Colony', 0.5046637058258057),
 ('Artificial_Intelligence_AAAI', 0.5035207867622375),
 ('Information_Retrieval', 0.5031781792640686),
 ('Neuroeconomics', 0.49684688448905945)]


vec = wv['modi'] + wv['Tata'] + wv['Reliance'] + wv['rich']
wv.most_similar([vec])

[('Reliance', 0.7896878123283386),
 ('Tata', 0.7583782076835632),
 ('Reliance_Industries', 0.6783670783042908),
 ('Tatas', 0.6754528880119324),
 ('RIL', 0.6713529825210571),
 ('Ambani', 0.664646327495575),
 ('Anil_Ambani', 0.6604036092758179),
 ('TATA', 0.6554300785064697),
 ('Bharti', 0.6534833908081055),
 ('Mukesh', 0.6437507271766663)]

#Comparing 2 objects

wv.similarity('man','women')
0.2883053

# Find odd in out
wv.doesnt_match(['tiger','lion','bottle','monkey'])
'bottle'

ans = (wv['king'] - wv['man']) + wv['women']
wv.most_similar([ans])
[('king', 0.6478992104530334),
 ('queen', 0.535493791103363),
 ('women', 0.52336585521698),
```

```
 ('kings', 0.5162314772605896),
 ('queens', 0.4995364844799042),
 ('kumaris', 0.492384672164917),
 ('princes', 0.46233269572257996),
 ('monarch', 0.4528028964996338),
 ('monarchy', 0.429317444562912),
 ('kings_princes', 0.42342400550842285)]

x = (wv['Man'] - wv['love'])
wv.most_similar([x])
[('Man', 0.7344020009040833),
 ('Woman', 0.5112388134002686),
 ('Capri_Pacing_Series', 0.41140201687812805),
 ('Suspect', 0.4090985953807831),
 ('Robber', 0.39425453543663025),
 ('Assailant', 0.3806186318397522),
 ('Panhandler', 0.3780188262462616),
 ('Capri_Casinos_ISLE', 0.37684619426727295),
 ('Capri_ISLE', 0.37642166018486023),
 ('Shoplifter', 0.3727079927921295)]
```

#Conclusion:

One-Hot Encoding provides a simple and sparse representation of categorical text data, useful for machine learning models with limited vocabulary. However, it lacks contextual depth. TF-IDF adds more value by weighting words based on their importance in the corpus, offering better distinction among terms. Additionally, using pre-trained word embeddings like Word2Vec enables high-dimensional representations that capture semantic relationships between words, which can enhance the performance of NLP models by providing a richer understanding of context and meaning.