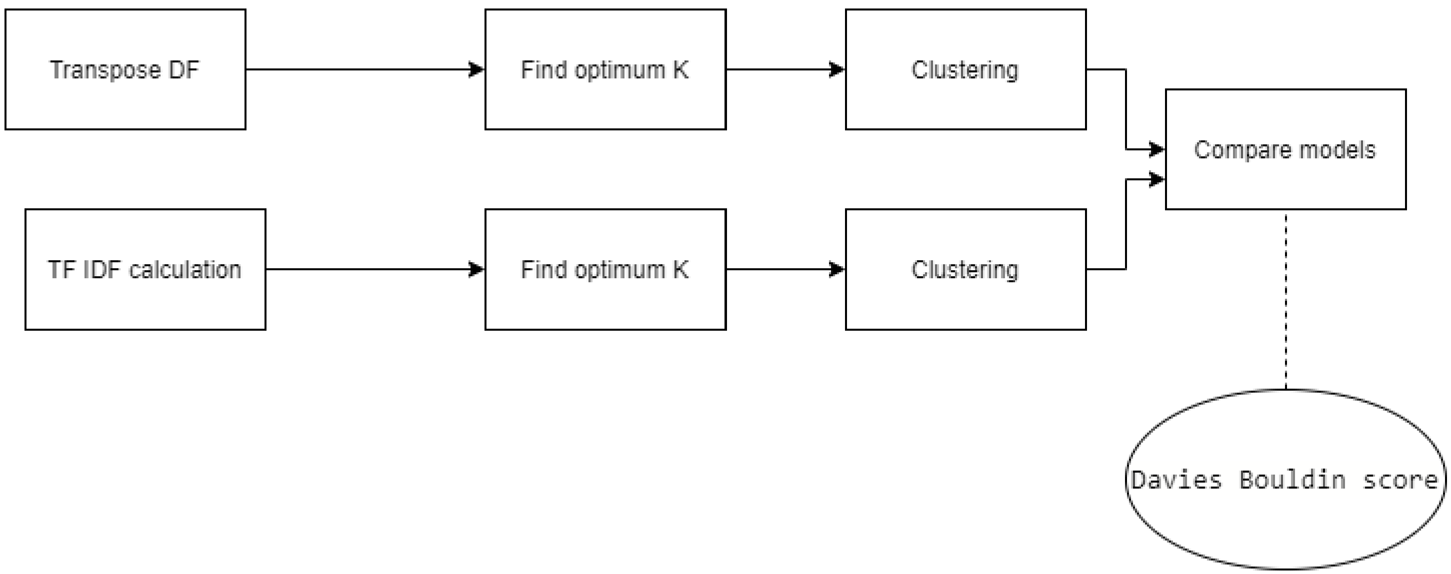
Methodology

- **Data Cleaning:** As per the data description the data has already been cleaned.
- **Data Visualization:** In this context for finding the optimal k value we can use data visualization.
- **Feature Selection:** Based on data description and
- **Model Selection:** K-means clustering is used in case of document clustering
- **Evalution:** Using inner analysis as actual clustreing values are not available.




```
import pandas as pd
import matplotlib.pyplot as plt
```

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remou
```

```
df = pd.read_csv("drive/My Drive/NIPS_1987-2015.csv")
```

```
df
```

	Unnamed: 0	1987_1	1987_2	1987_3	1987_4	1987_5	1987_6	1987_7	1987_8	1987_9	1987_10	1987_11	1987_12	19
0	abalone	0	0	0	0	0	0	0	0	0	0	0	0	
1	abbeel	0	0	0	0	0	0	0	0	0	0	0	0	
2	abbott	0	0	0	0	0	0	0	0	0	0	0	0	
3	abbreviate	0	0	0	0	0	0	0	0	0	0	0	0	
4	abbreviated	0	0	0	0	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
11458	zoo	0	0	0	0	0	0	0	0	0	0	0	0	
11459	zoom	0	0	0	0	0	0	0	0	0	0	0	0	
11460	zou	0	0	0	0	0	0	0	0	0	0	0	0	
11461	zoubin	0	0	0	0	0	0	0	0	0	0	0	0	
11462	zurich	0	0	0	0	0	0	0	0	0	0	0	0	

11463 rows × 5812 columns

The data is a collection of words in papers and their frequency in 5812 papers in years 1987 - 2015

```
c=list(df)
words = list(df[c[0]])
df.drop(c[0],axis=1,inplace=True)
```

we can drop the word because only the count of the word is important than what is word itself is.

Clustering on Count of The word

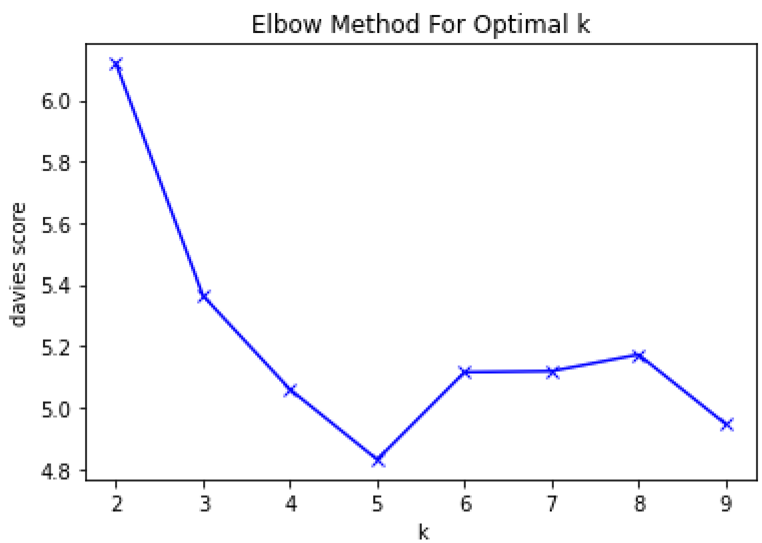
```
df=df.T
df
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1987_1	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1987_2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0
1987_3	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1987_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1987_5	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2015_399	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
2015_400	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
2015_401	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
2015_402	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
2015_403	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0

5811 rows × 11463 columns

Transpsosing the data as each documnet is a single row giving count of different words. This is the format need to be used in case of document clustering.

```
from sklearn.cluster import KMeans
Sum_of_squared_distances = []
from sklearn.metrics import davies_bouldin_score
K = range(2,10)
for k in K:
   km = KMeans(n_clusters=k, max_iter=200, n_init=10)
   km = km.fit(df)
   Sum_of_squared_distances.append(davies_bouldin_score(df, km.labels_))
plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('davies score')
plt.title('Elbow Method For Optimal k')
plt.show()
```



Finding optimal k value

```
from sklearn.cluster import KMeans
num_clusters = Sum_of_squared_distances.index(min(Sum_of_squared_distances))+2
km = KMeans(n_clusters=num_clusters)
km.fit(df)
clusters = km.labels_.tolist()
```

Fitting model on count of of words

```
docs = c[1:]
frame1=pd.DataFrame(docs , columns=['docName'])
```

```
frame1['clusterid'] = clusters
frame1
```

	docName	clusterid
0	1987_1	4
1	1987_2	4
2	1987_3	4
3	1987_4	4
4	1987_5	4
...	...	...
5806	2015_399	2
5807	2015_400	2
5808	2015_401	2
5809	2015_402	2
5810	2015_403	1

5811 rows × 2 columns

```
labels = km.labels_
davies_bouldin_score(df, labels)
```

```
4.850590228382116
```

davies bouldin score for evaluating clustering. The minimum value will be 0.

`Clustering on TF-IDF index of the word

TD-IDF is a statistical method used in NLP for document clustering,which will tell the importance of a word in a document. The term frequencymatrixis created using https://www.tfidf.com/

```
import numpy as np
doc_term_matrix=df.to_numpy()
```

```
def tfCalcForDoc(doc_term):
    docTfMat=[]
    totalWordCnt=np.sum(doc_term)
    for termCnt in doc_term:
        if totalWordCnt != 0:
            docTfMat.append(termCnt/totalWordCnt)
        else:
            docTfMat.append(0)
    return docTfMat
```

calculating TF of the document

```
doc_freq_array = []
for doc_term in doc_term_matrix:
    doc_freq_array.append(tfCalcForDoc(doc_term))
```

```
doc_freq_matrix=np.asmatrix(doc_freq_array)
```

```
df=df.T
df.shape
```

```
(11463, 5811)
```

```
def idfCalcForTerm(term_count_array_across_docs):
    # Total number of documents
    nr = len(term_count_array_across_docs)
    # Number of documents with term t in it
    dr = np.count_nonzero(term_count_array_across_docs)

    if dr != 0:
        return np.log10(nr/dr)
    else:
        return 0
```

calculating idf ofthe document

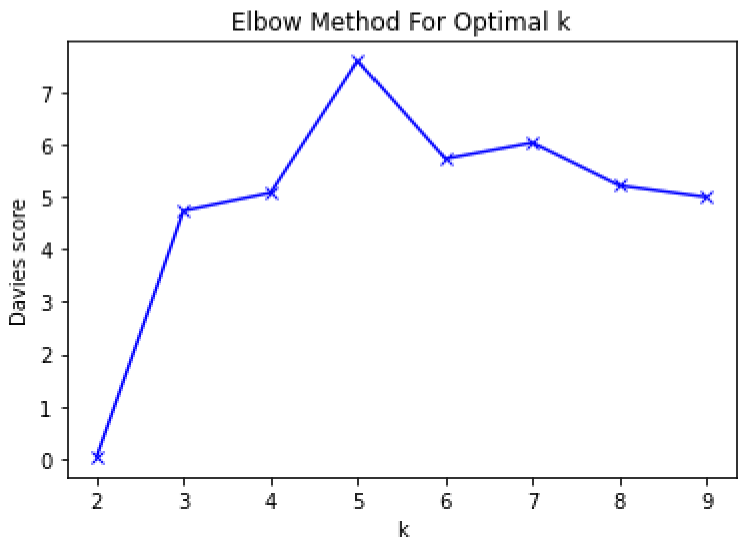
```
arr=np.squeeze(np.asarray(df))
idfArr=[]
for m in arr:
    idfArr.append(idfCalcForTerm(m))
```

```
cnt=0
for idfVals in idfArr:
    doc_freq_matrix[:,cnt] *= idfVals
    cnt+= 1
```

idf factor calculation

```
Sum_of_squared_distances = []
K = range(2,10)
for k in K:
   km = KMeans(n_clusters=k)
   km = km.fit(doc_freq_matrix)
   Sum_of_squared_distances.append(davies_bouldin_score(doc_freq_matrix, km.labels_))
```

```
plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Davies score')
plt.title('Elbow Method For Optimal k')
plt.show()
```



finding oprimal k

```
from sklearn.cluster import KMeans
num_clusters = Sum_of_squared_distances.index(min(Sum_of_squared_distances))+2
km = KMeans(n_clusters=num_clusters)
km.fit(doc_freq_matrix)
clusters = km.labels_.tolist()
```

```
docs = c[1:]
frame2=pd.DataFrame(docs , columns=['docName'])
frame2['clusterid'] = clusters
frame2
```

	docName	clusterid
0	1987_1	0
1	1987_2	0
2	1987_3	0
3	1987_4	0
4	1987_5	0
...	...	...
5806	2015_399	0
5807	2015_400	0
5808	2015_401	0
5809	2015_402	0
5810	2015_403	0

```
labels = km.labels_
from sklearn.metrics import davies_bouldin_score
davies_bouldin_score(doc_freq_matrix, labels)
```

```
0.03564458840070297
```

Comaparing Both methods using davies bouldin score.

This is an internal evaluation scheme, where the validation of how well the clustering has been done is made using quantities and features inherent to the dataset.

So, TF-IDF method has better feature information than counting values