In [11]:
```
import pandas as pd
import nltk
nltk.download("stopwords")
from nltk.corpus import stopwords
nltk.download("punkt")
from snowballstemmer import TurkishStemmer
from textblob import TextBlob
from sklearn import model_selection, preprocessing, naive_bayes, metrics
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn import decomposition, ensemble
import numpy, string


# ============================================================================
=
# Read Train Data
# ============================================================================
=
index_names=[]
RawDataFrame=[]
for i in [i for i in range(1,751) if i<451 or i>600]:
    file = open("train/{}.txt".format(i), "r", encoding='iso8859-9')#turkishco
de
    index_names.append("{}.txt".format(i))
    filetext=file.read()
    file.close()
    RawDataFrame.append(filetext)

# ============================================================================
=
# Read Test Data
# ============================================================================
=
for i in [i for i in range(1,401) if i<241 or i>320]:
    file = open("test/testdat {}.txt".format(i), "r", encoding='iso8859-9')#tu
rkishcode
    index_names.append("testdat {}.txt".format(i))
    filetext=file.read()
    file.close()
    RawDataFrame.append(filetext)

# ============================================================================
=
# Convert Data Structer which i want style
# ============================================================================
=
df=pd.Series(RawDataFrame)

# ============================================================================
=
# Cleaning about lowercase and punction and number
# ============================================================================
=
df=df.apply(lambda x: " ".join(x.lower() for x in x.split()))
df=df.replace('[^\w\s]','',regex=True)#regex
df=df.replace('\d+','',regex=True)#regex
```

```
# ================================================================================
=
# Cleaning StopWords
# ================================================================================
=
sw = stopwords.words("turkish")
df=df.apply(lambda x: " ".join(x for x in x.split() if x not in sw))

# ================================================================================
=
# If its frequency is 1 inside all data, it be requed to remove because of Fea
ture Selection
# ================================================================================
=

#pd.Series(" ".join(df["News"]).split()).value_counts()

# ================================================================================
=
# Tokenizing
# ================================================================================
=
df=df.apply(lambda x: TextBlob(x).words)

# ================================================================================
=
# Stemming
# ================================================================================
=
stemmer = TurkishStemmer()
df=df.apply(lambda x: " ".join(stemmer.stemWord(word) for word in x))

# ================================================================================
=
# AddingClass   0 ekonomi     1 magazin     2 saglik     3 spor
# ================================================================================
=
Category=["ekonomi" for i in range(150)]
Category.extend(["magazin" for i in range(150)])
Category.extend(["saglik" for i in range(150)])
Category.extend(["spor" for i in range(150)])
Category.extend(["ekonomi" for i in range(80)])
Category.extend(["magazin" for i in range(80)])
Category.extend(["saglik" for i in range(80)])
Category.extend(["spor" for i in range(80)])

dframe=pd.DataFrame(df,columns=["News"])
dframe=dframe.assign(category=Category)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\suca\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\suca\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [12]:
```
train_x,train_y,test_x,test_y=dframe.iloc[:600,[0]].values.ravel(),dframe.iloc
[:600,[1]].values.ravel(),dframe.iloc[600:,[0]].values.ravel(),dframe.iloc[600
:,[1]].values.ravel()
encoder = preprocessing.LabelEncoder() #Prepare Encoder
train_y = encoder.fit_transform(train_y)
test_y = encoder.fit_transform(test_y)
```

WORDLEWEL

In [13]:
```
tfidf_vectorizer=TfidfVectorizer()
tfidf_vectorizer.fit(train_x)
x_train_tfidf= tfidf_vectorizer.transform(train_x)
x_test_tfidf = tfidf_vectorizer.transform(test_x)


#tfidf_df=pd.DataFrame(x_train_tfidf.toarray())
#tfidf_df.insert(0, 'News_Number', index_names)
#tfidf_df["Sinif"]=dframe.iloc[:600,[1]].values.ravel()

#tfidf_df.to_csv("Result.txt")
```

Bayes

In [15]:
```
nb = naive_bayes.MultinomialNB()
nb_model = nb.fit(x_train_tfidf,train_y)
accuracy = model_selection.cross_val_score(nb_model,
                                           x_test_tfidf,
                                           test_y,
                                           cv = 10).mean()

print("Doğruluk Oranı:", accuracy)
```

```
Doğruluk Oranı: 0.98125
```

In [16]:
```
Results=pd.DataFrame(metrics.precision_recall_fscore_support(nb_model.predict(
x_test_tfidf.toarray()),test_y),columns=['Ekonomi','Magazin','Saglik','Spor'])
Results["Average"]=Results.mean(axis=1)
Results.insert(0, ' ', ['Precision','Recall','Fbeta_Score','Support'])
Results
```

Out[16]:

		Ekonomi	Magazin	Saglik	Spor	Average
0	Precision	1.000000	0.962500	0.962500	0.9875	0.978125
1	Recall	0.952381	1.000000	0.974684	0.9875	0.978641
2	Fbeta_Score	0.975610	0.980892	0.968553	0.9875	0.978139
3	Support	84.000000	77.000000	79.000000	80.0000	80.000000

#KNN

In [19]:
```
from sklearn.neighbors import KNeighborsClassifier as knn
knn = knn()
knn_model = knn.fit(x_train_tfidf,train_y)
accuracy = model_selection.cross_val_score(knn_model,
                                           x_test_tfidf,
                                           test_y,
                                           cv = 10).mean()

print("Doğruluk Oranı:", accuracy)# DAha Dusuk
```

```
Doğruluk Oranı: 0.95
```

In [ ]:
```

```