In [14]:
```
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import ComplementNB

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

In [15]:
```
df = pd.read_csv('CaloriesIntensityUpload2.csv')
```

In [16]:
```
df
```

Out[16]:

	Id	Time	Intensity	Calories	DateHour	Date	TimeofDay
0	1503960366	0:00:00	10	66	2016-04-24T00:00:00Z	4/24/2016	Night
1	1503960366	0:00:00	1	51	2016-04-30T00:00:00Z	4/30/2016	Night
2	1624580081	0:00:00	1	51	2016-04-14T00:00:00Z	4/14/2016	Night
3	1624580081	0:00:00	1	51	2016-04-21T00:00:00Z	4/21/2016	Night
4	1624580081	0:00:00	1	51	2016-04-22T00:00:00Z	4/22/2016	Night
...	...	...	...	...	...	...	...
22094	5553957443	22:00:00	25	114	2016-05-10T22:00:00Z	5/10/2016	Evening
22095	7086361926	22:00:00	20	114	2016-04-30T22:00:00Z	4/30/2016	Evening
22096	1644430081	23:00:00	8	114	2016-04-16T23:00:00Z	4/16/2016	Night
22097	4445114986	23:00:00	22	114	2016-04-28T23:00:00Z	4/28/2016	Night
22098	8583815059	23:00:00	9	114	2016-04-12T23:00:00Z	4/12/2016	Night

22099 rows × 7 columns

In [17]:
```
df.Calories.value_counts()
```

Out[17]:
```
86     1068
56     1060
84      888
62      711
83      697
       ...
506       1
364       1
394       1
741       1
512       1
Name: Calories, Length: 442, dtype: int64
```

In [18]:
```
tfidfv = TfidfVectorizer()
tfidf_ngram_features = tfidfv.fit_transform(df['TimeofDay'])
tfidf_ngram_features
```

Out[18]:
```
<22099x4 sparse matrix of type '<class 'numpy.float64'>'
        with 22099 stored elements in Compressed Sparse Row format>
```

In [19]:
```
countvec = CountVectorizer()
countvec_ngram_features = countvec.fit_transform(df['TimeofDay'])
countvec_ngram_features
```

Out[19]:
```
<22099x4 sparse matrix of type '<class 'numpy.int64'>'
        with 22099 stored elements in Compressed Sparse Row format>
```

In [20]:
```
# TFIDF + KNC
X_train, X_test, y_train, y_test = train_test_split(tfidf_ngram_features, df['TimeofDay'], test_size=0.3, rando
model = KNeighborsClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred, digits=4, target_names=list(map(str, list(y_test.unique())))))
```

```
              precision    recall  f1-score   support

       Night     1.0000    1.0000    1.0000      1569
     Evening     1.0000    1.0000    1.0000      1648
   Afternoon     1.0000    1.0000    1.0000      1681
     Morning     1.0000    1.0000    1.0000      1732

    accuracy                         1.0000      6630
   macro avg     1.0000    1.0000    1.0000      6630
weighted avg     1.0000    1.0000    1.0000      6630
```

In [21]:
```
# TFIDF + CNB
```

```
X_train, X_test, y_train, y_test = train_test_split(tfidf_ngram_features, df['TimeofDay'], test_size=0.3, rando
model = ComplementNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred, digits=4, target_names=list(map(str, list(y_test.unique())))))
```

```
              precision    recall  f1-score   support

       Night     1.0000    1.0000    1.0000      1569
     Evening     1.0000    1.0000    1.0000      1648
   Afternoon     1.0000    1.0000    1.0000      1681
     Morning     1.0000    1.0000    1.0000      1732

    accuracy                         1.0000      6630
   macro avg     1.0000    1.0000    1.0000      6630
weighted avg     1.0000    1.0000    1.0000      6630
```

In [22]:
```
# CountVec + CNB
X_train, X_test, y_train, y_test = train_test_split(countvec_ngram_features, df['TimeofDay'],
                                                    test_size=0.3, random_state=1)
model = ComplementNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred, digits=4, target_names=list(map(str, list(y_test.unique())))))
```

```
              precision    recall  f1-score   support

       Night     1.0000    1.0000    1.0000      1569
     Evening     1.0000    1.0000    1.0000      1648
   Afternoon     1.0000    1.0000    1.0000      1681
     Morning     1.0000    1.0000    1.0000      1732

    accuracy                         1.0000      6630
   macro avg     1.0000    1.0000    1.0000      6630
weighted avg     1.0000    1.0000    1.0000      6630
```

In [23]:
```
# CountVec + CNB
X_train, X_test, y_train, y_test = train_test_split(countvec_ngram_features, df['TimeofDay'],
                                                    test_size=0.3, random_state=1)
model = ComplementNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred, digits=4, target_names=list(map(str, list(y_test.unique())))))
```

```
              precision    recall  f1-score   support

       Night     1.0000    1.0000    1.0000      1569
     Evening     1.0000    1.0000    1.0000      1648
   Afternoon     1.0000    1.0000    1.0000      1681
     Morning     1.0000    1.0000    1.0000      1732

    accuracy                         1.0000      6630
   macro avg     1.0000    1.0000    1.0000      6630
weighted avg     1.0000    1.0000    1.0000      6630
```

In [ ]: