In [71]:
```
#Importing necessary libraries
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.ticker as mtick
import matplotlib.pyplot as plt
import seaborn as sns
import torch
from sklearn.metrics import classification_report
import torch.nn as nn
from sklearn.neural_network import MLPRegressor
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
import time
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
from sklearn.model_selection import train_test_split, StratifiedKFold
from joblib import dump
import pickle
from sklearn.metrics import accuracy_score
from joblib import load
from sklearn.metrics import confusion_matrix
```

Read Test Data

In [72]:
```
#TF-IDF
x_test_tf_idf = "Data/x_test_tf_idf.pickle"
with open(x_test_tf_idf, 'rb') as x:
    x_test_tf_idf = pickle.load(x)

y_test_tf = "Data/y_test_tf.pickle"
with open(y_test_tf, 'rb') as x:
    y_test_tf = pickle.load(x)

#TF-IDF with stop
x_test_tf_idf_stop = "Data/x_test_tf_idf_stop.pickle"
with open(x_test_tf_idf_stop, 'rb') as x:
    x_test_tf_idf_stop = pickle.load(x)

y_test_tf_stop = "Data/y_test_tf_stop.pickle"
with open(y_test_tf_stop, 'rb') as x:
    y_test_tf_stop = pickle.load(x)

#W2V
x_test_w2v = "Data/x_test_w2v.pickle"
with open(x_test_w2v, 'rb') as x:
    x_test_w2v = pickle.load(x)

y_test_w2v = "Data/y_test_w2v.pickle"
with open(y_test_w2v, 'rb') as x:
    y_test_w2v = pickle.load(x)

#W2V with stop
x_test_w2v_stop = "Data/x_test_w2v_stop.pickle"
with open(x_test_w2v_stop, 'rb') as x:
    x_test_w2v_stop = pickle.load(x)

y_test_w2v_stop = "Data/y_test_w2v_stop.pickle"
with open(y_test_w2v_stop, 'rb') as x:
    y_test_w2v_stop = pickle.load(x)
```

Testing on Baseline model (Naive Bayes + TF-IDF)

In [73]:
```
#Load model
nb = load('01_NB.joblib')
nb_stop = load('01_NB_stop.joblib')
```

NB + TF-IDF without stopwords

In [74]:
```
start_time = time.time() #Measure time
pred = nb.predict(x_test_tf_idf) #Test model
testing_time = time.time() - start_time #Testing time

#Print testing accuracy
print("Testing accuracy for Naive Bayes without stopwords: ", accuracy_score(y
#Print testing time
print("Testing time for Naive Bayes without stopwords:: ", testing_time)
```

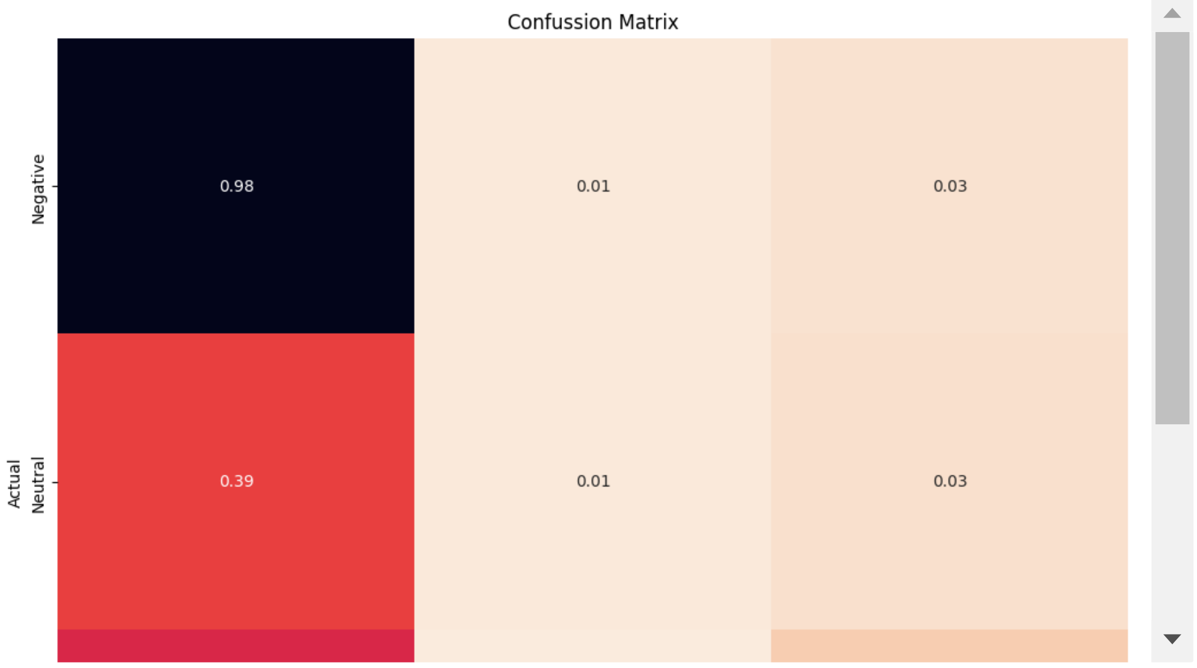
```
Testing accuracy for Naive Bayes without stopwords:  54.0021731256791
Testing time for Naive Bayes without stopwords::  0.005984306335449219
```

Confusion Matrix

In [75]:
```
print(classification_report(y_test_tf, pred))
```

```
              precision    recall  f1-score   support

    Negative       0.54      0.98      0.70      1451
     Neutral       0.50      0.01      0.02       597
    Positive       0.59      0.08      0.14       713

    accuracy                           0.54      2761
   macro avg       0.54      0.36      0.29      2761
weighted avg       0.54      0.54      0.41      2761
```

In [78]:
```
fig = plt.figure(figsize = (12,10)) #Define figure size
mat = confusion_matrix(y_test_tf, pred) #Define confussion matrix with test da
mat = mat/np.sum(mat, axis = 1) #Normalizing
name = ['Negative', 'Neutral', 'Positive'] #Define names of the class accordin

#Heatmap
sns.heatmap(mat, annot = True, cmap = 'rocket_r', fmt = '.2f',xticklabels=name
plt.title('Confussion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')

plt.show()
```



In [ ]:

NB + TF-IDF with stopwords

In [79]:
```
start_time = time.time() #Measure time
pred = nb_stop.predict(x_test_tf_idf_stop) #Test model
testing_time = time.time() - start_time #Testing time

#Print testing accuracy
print("Testing accuracy for Naive Bayes without stopwords: ", accuracy_score(y
#Print testing time
print("Testing time for Naive Bayes without stopwords:: ", testing_time)
```

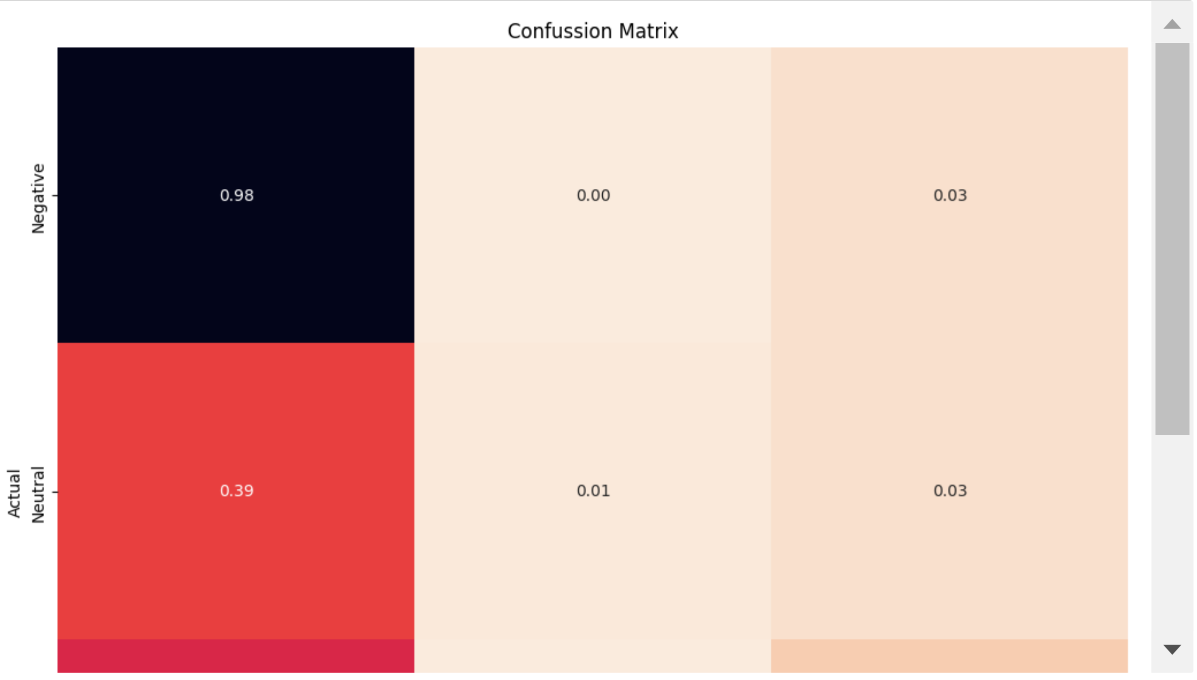
```
Testing accuracy for Naive Bayes without stopwords:  53.965954364360734
Testing time for Naive Bayes without stopwords::  0.0019638538360595703
```

Confusion Matrix

In [80]:
```
print(classification_report(y_test_tf, pred))
```

```
              precision    recall  f1-score   support

    Negative       0.54      0.98      0.70      1451
     Neutral       0.62      0.01      0.02       597
    Positive       0.57      0.08      0.14       713

    accuracy                           0.54      2761
   macro avg       0.58      0.36      0.28      2761
weighted avg       0.57      0.54      0.41      2761
```

In [81]:
```
fig = plt.figure(figsize = (12,10)) #Define figure size
mat = confusion_matrix(y_test_tf, pred) #Define confussion matrix with test da
mat = mat/np.sum(mat, axis = 1) #Normalizing
name = ['Negative', 'Neutral', 'Positive'] #Define names of the class accordin

#Heatmap
sns.heatmap(mat, annot = True, cmap = 'rocket_r', fmt = '.2f',xticklabels=name
plt.title('Confussion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')

plt.show()
```



Testing on MLP

MLP+W2V

In [82]:
```
#Load model
ml = load('02_MLP_Optimised_w2v.joblib')
```

In [83]:
```
start_time = time.time() #Measure time
pred3 = ml.predict(x_test_w2v) #Test model
testing_time = time.time() - start_time #Testing time

#Print testing accuracy
print("Testing accuracy for MLP+W2V: ", accuracy_score(y_test_w2v, pred3)*100)
#Print testing time
print("Testing time for MLP+W2V: ", testing_time)
```

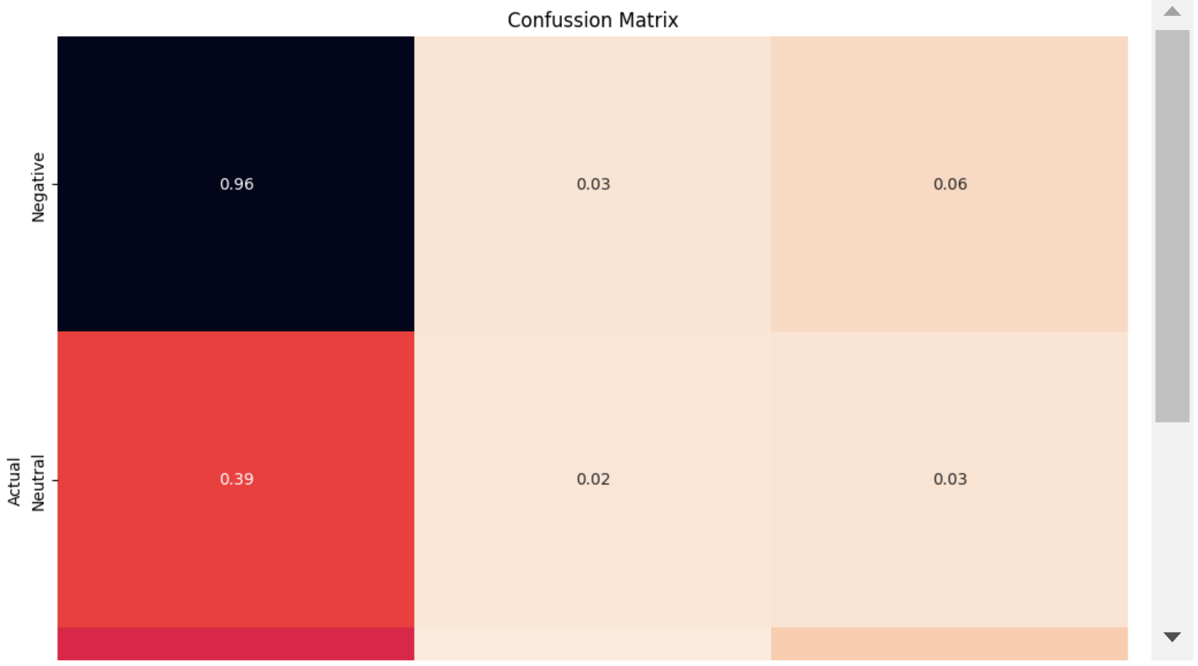
```
Testing accuracy for MLP+W2V:  53.38645418326693
Testing time for MLP+W2V:  0.012926101684570312
```

Confusion Matrix

In [84]:
```
print(classification_report(y_test_w2v, pred3))
```

```
              precision    recall  f1-score   support

    Negative       0.54      0.96      0.69      1451
     Neutral       0.40      0.02      0.04       597
    Positive       0.51      0.09      0.15       713

    accuracy                           0.53      2761
   macro avg       0.48      0.36      0.30      2761
weighted avg       0.50      0.53      0.41      2761
```

In [85]:
```
fig = plt.figure(figsize = (12,10)) #Define figure size
mat = confusion_matrix(y_test_tf, pred3) #Define confussion matrix with test d
mat = mat/np.sum(mat, axis = 1) #Normalizing
name = ['Negative', 'Neutral', 'Positive'] #Define names of the class accordin

#Heatmap
sns.heatmap(mat, annot = True, cmap = 'rocket_r', fmt = '.2f',xticklabels=name
plt.title('Confussion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')

plt.show()
```



MLP+TF-IDF

In [86]:
```
#Load model
ml2 = load('02_MLP_Optimised_tf_idf.joblib')
```

In [87]:
```
start_time = time.time() #Measure time
pred4 = ml2.predict(x_test_tf_idf) #Test model
testing_time = time.time() - start_time #Testing time

#Print testing accuracy
print("Testing accuracy for MLP+TF-IDF: ", accuracy_score(y_test_tf, pred4)*10
#Print testing time
print("Testing time for MLP+TF-IDF: ", testing_time)
```

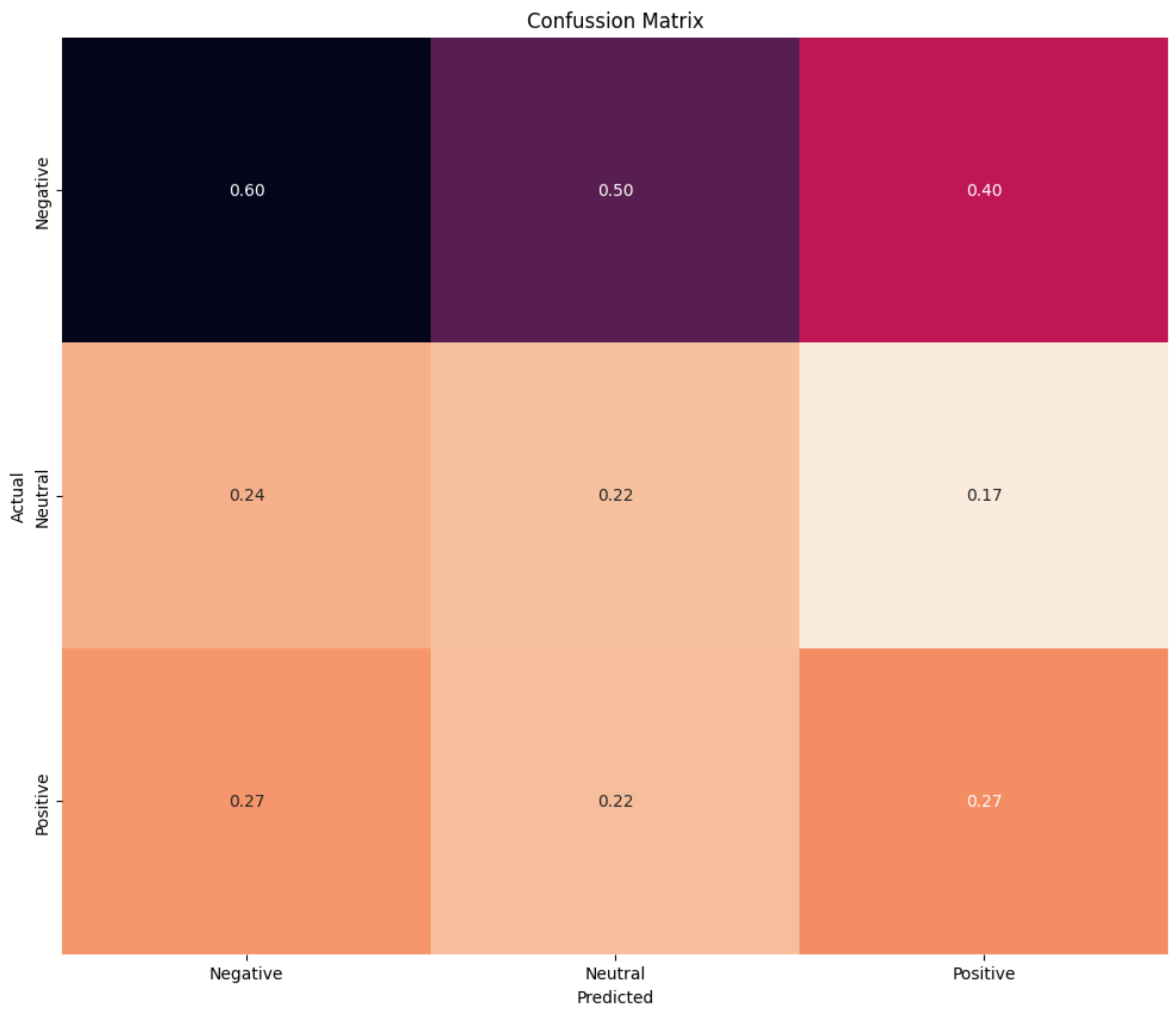
```
Testing accuracy for MLP+TF-IDF:  43.245201014125314
Testing time for MLP+TF-IDF:  0.0029752254486083984
```

Confusion Matrix

```
In [88]: print(classification_report(y_test_tf, pred4))
```

```
              precision    recall  f1-score   support

    Negative       0.54      0.60      0.57      1451
     Neutral       0.23      0.22      0.23       597
    Positive       0.32      0.27      0.30       713

    accuracy                           0.43      2761
   macro avg       0.37      0.36      0.36      2761
weighted avg       0.42      0.43      0.42      2761
```

In [89]:
```
fig = plt.figure(figsize = (12,10)) #Define figure size
mat = confusion_matrix(y_test_tf, pred4) #Define confussion matrix with test a
mat = mat/np.sum(mat, axis = 1) #Normalizing
name = ['Negative', 'Neutral', 'Positive'] #Define names of the class accordin

#Heatmap
sns.heatmap(mat, annot = True, cmap = 'rocket_r', fmt = '.2f',xticklabels=name
plt.title('Confussion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')

plt.show()
```



Misclassification (MLP+TF-IDF)

In [126]:
```
y_test_tf_r = y_test_tf.reset_index(drop=True)
misclassified = []
for i in range(len(y_test_tf_r)):
    if y_test_tf_r[i] != pred4[i]:
        misclassified.append((i, pred4[i]))
```

In [127]:
```
y_test_tf_r = y_test_tf.reset_index(drop=True)
mis_lis = []
for i in range(len(y_test_tf_r)):
    if y_test_tf_r[i] != pred4[i]:
        mis_lis.append((i))
```

In [123]:
```
mis = df = pd.read_csv('Data/my_data.csv')
```

In [124]:
```
mis
```

Out[124]:

	Unnamed: 0	comment	annotation	clean_sentence	tokenized_text	tokenized_text_no_
0	0	লিখার সময় পারলে সত্য লিখার অভ্যাস শিখুন।	Negative	লিখার সময় পারলে সত্য লিখার অভ্যাস শিখুন	['লিখার', 'সময়', 'পারলে', 'সত্য', 'লিখার', 'অভ...	['লিখার', 'সময়', 'পা 'সত্য', 'লিখার', '
1	1	এটা কেন হচ্ছে? সংশ্লিষ্ট সকলের ডিপ্রেশনের ফলে?...	Negative	এটা কেন হচ্ছে সংশ্লিষ্ট সকলের ডিপ্রেশনের ফলে ন...	['এটা', 'কেন', 'হচ্ছে', 'সংশ্লিষ্ট', 'সকলের', ...	['সংশ্লিষ্ট', 'সক 'ডিপ্রেশনের', 'সরকা
2	2	আমাদের দেশের স্বাভাবিক অর্থনৈতিক	Negative	আমাদের দেশের স্বাভাবিক অর্থনৈতিক	['আমাদের', 'দেশের', 'স্বাভাবিক',	['দেশের', 'স্বাভা 'অর্থনৈতিক', 'গতিপ্র

In [128]:
```
mis = mis.loc[mis_lis]
```

In [130]:
```
misclassified
```

Out[130]:
```
[(0, 'Neutral'),
 (6, 'Neutral'),
 (9, 'Neutral'),
 (10, 'Neutral'),
 (12, 'Negative'),
 (13, 'Negative'),
 (14, 'Positive'),
 (15, 'Negative'),
 (18, 'Neutral'),
 (19, 'Negative'),
 (20, 'Neutral'),
 (23, 'Neutral'),
 (24, 'Positive'),
 (25, 'Neutral'),
 (27, 'Neutral'),
 (30, 'Neutral'),
 (33, 'Negative'),
 (34, 'Negative'),
 (35, 'Negative'),
```

In [131]:
```
mis.head(10)
```

Out[131]:

	Unnamed: 0	comment	annotation	clean_sentence	tokenized_text	tokenized_text_no_stop	t
0	0	লিখার সময় পারলে সত্য লিখার অভ্যাস শিখুন।	Negative	লিখার সময় পারলে সত্য লিখার অভ্যাস শিখুন	['লিখার', 'সময়', 'পারলে', 'সত্য', 'লিখার', 'অভ...	['লিখার', 'সময়', 'পারলে', 'সত্য', 'লিখার', 'অভ...	
6	6	সরকার যাদের এই ব্যাংকে নিয়গ দিয়েছে তারা ব্যাংক...	Negative	সরকার যাদের এই ব্যাংকে নিয়গ দিয়েছে তারা ব্যাংক...	['সরকার', 'যাদের', 'এই', 'ব্যাংকে', 'নিয়গ', 'দ...	['সরকার', 'ব্যাংকে', 'নিয়গ', 'দিয়েছে', 'ব্যাংক...	
9	9	ইসলামি ব্যাংক প্রারম্ভ থেকেই গ্রাহকদের পছন্দের...	Negative	ইসলামি ব্যাংক প্রারম্ভ থেকেই গ্রাহকদের পছন্দের...	['ইসলামি', 'ব্যাংক', 'প্রারম্ভ', 'থেকেই', 'গ্র...	['ইসলামি', 'ব্যাংক', 'প্রারম্ভ', 'গ্রাহকদের', ...	
10	10	এরা যেখানেই যাবে সেখানেই চুরি হবে।	Negative	এরা যেখানেই যাবে সেখানেই চুরি হবে	['এরা', 'যেখানেই', 'যাবে', 'সেখানেই', 'চুরি', ...	['যেখানেই', 'সেখানেই', 'চুরি']	
12	12	শেয়ার প্রতি আয় কমেছে। শূন্য হয় নি এখনও। সরকারী...	Negative	শেয়ার প্রতি আয় কমেছে শূন্য হয় নি এখনও সরকারী [...	['শেয়ার', 'প্রতি', 'আয়', 'কমেছে', 'শূন্য', 'হয়...	['শেয়ার', 'আয়', 'কমেছে', 'শূন্য', 'নি', 'সরকার...	
13	13	পুরোপুরি জামাতমুক্ত করা গেলেই লাভের মুখ দেখবে ...	Positive	পুরোপুরি জামাতমুক্ত করা গেলেই লাভের মুখ দেখবে ...	['পুরোপুরি', 'জামাতমুক্ত', 'করা', 'গেলেই', 'লা...	['পুরোপুরি', 'জামাতমুক্ত', 'গেলেই', 'লাভের', '...	
14	14	বিগত কয়েক বছরের অভিজ্ঞতা বলে ব্যাংকসহ শ্যেনদৃষ...	Negative	বিগত কয়েক বছরের অভিজ্ঞতা বলে ব্যাংকসহ শ্যেনদৃষ...	['বিগত', 'কয়েক', 'বছরের', 'অভিজ্ঞতা', 'বলে', '...	['বিগত', 'বছরের', 'অভিজ্ঞতা', 'ব্যাংকসহ', 'শ্য...	
15	15	দ্রুত জামাত প্রভাব মুক্ত করা হউক, গ্রাহকরা উপক...	Neutral	দ্রুত জামাত প্রভাব মুক্ত করা হউক গ্রাহকরা উপকৃ...	['দ্রুত', 'জামাত', 'প্রভাব', 'মুক্ত', 'করা', '...	['দ্রুত', 'জামাত', 'প্রভাব', 'মুক্ত', 'হউক', '...	
18	18	প্রবাসী বাংলাদেশিরা চলতি অর্থবছরের (২০১৮-১৯) প...	Negative	প্রবাসী বাংলাদেশিরা চলতি অর্থবছরের প্রথম মাসে ...	['প্রবাসী', 'বাংলাদেশিরা', 'চলতি', 'অর্থবছরের'...	['প্রবাসী', 'বাংলাদেশিরা', 'চলতি', 'অর্থবছরের'...	

	Unnamed: 0	comment	annotation	clean_sentence	tokenized_text	tokenized_text_no_stop	t
19	19	গরীবদের বেশী টানটান হতে নাই।	Negative	গরীবদের বেশী টানটান হতে নাই	['গরীবদের', 'বেশী', 'টানটান', 'হতে', 'নাই']	['গরীবদের', 'বেশী', 'টানটান']	

In [ ]:

Testing on Decision Tree

Decision Tree+W2V

In [90]:
```
#Load model
dt = load('02_DT_Optimised_w2v.joblib')
```

In [91]:
```
start_time = time.time() #Measure time
pred5 = dt.predict(x_test_w2v) #Test model
testing_time = time.time() - start_time #Testing time

#Print testing accuracy
print("Testing accuracy for DT+W2V: ", accuracy_score(y_test_w2v, pred5)*100)
#Print testing time
print("Testing time for DT+W2V: ", testing_time)
```

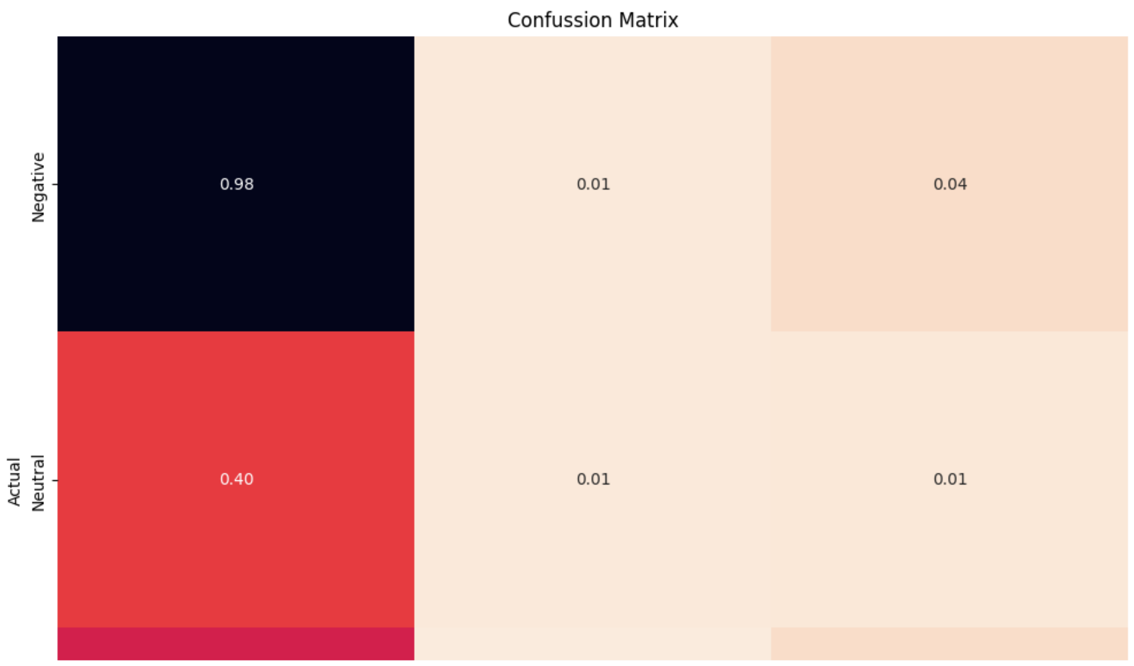
```
Testing accuracy for DT+W2V:  52.589641434262944
Testing time for DT+W2V:  0.00394439697265625
```

Confusion Matrix

In [92]:
```
print(classification_report(y_test_w2v, pred5))
```

```
              precision    recall  f1-score   support

    Negative       0.53      0.98      0.69      1451
     Neutral       0.46      0.01      0.02       597
    Positive       0.42      0.04      0.07       713

    accuracy                           0.53      2761
   macro avg       0.47      0.34      0.26      2761
weighted avg       0.49      0.53      0.38      2761
```

In [93]:
```
fig = plt.figure(figsize = (12,10)) #Define figure size
mat = confusion_matrix(y_test_tf, pred5) #Define confussion matrix with test d
mat = mat/np.sum(mat, axis = 1) #Normalizing
name = ['Negative', 'Neutral', 'Positive'] #Define names of the class accordin

#Heatmap
sns.heatmap(mat, annot = True, cmap = 'rocket_r', fmt = '.2f',xticklabels=name
plt.title('Confussion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')

plt.show()
```



Decision Tree+TF-IDF

In [94]:
```
#Load model
dt2 = load('02_DT_Optimised_tf_idf.joblib')
```

In [95]:
```
start_time = time.time() #Measure time
pred6 = dt2.predict(x_test_tf_idf) #Test model
testing_time = time.time() - start_time #Testing time

#Print testing accuracy
print("Testing accuracy for DT+W2V: ", accuracy_score(y_test_tf, pred6)*100)
#Print testing time
print("Testing time for DT+W2V: ", testing_time)
```

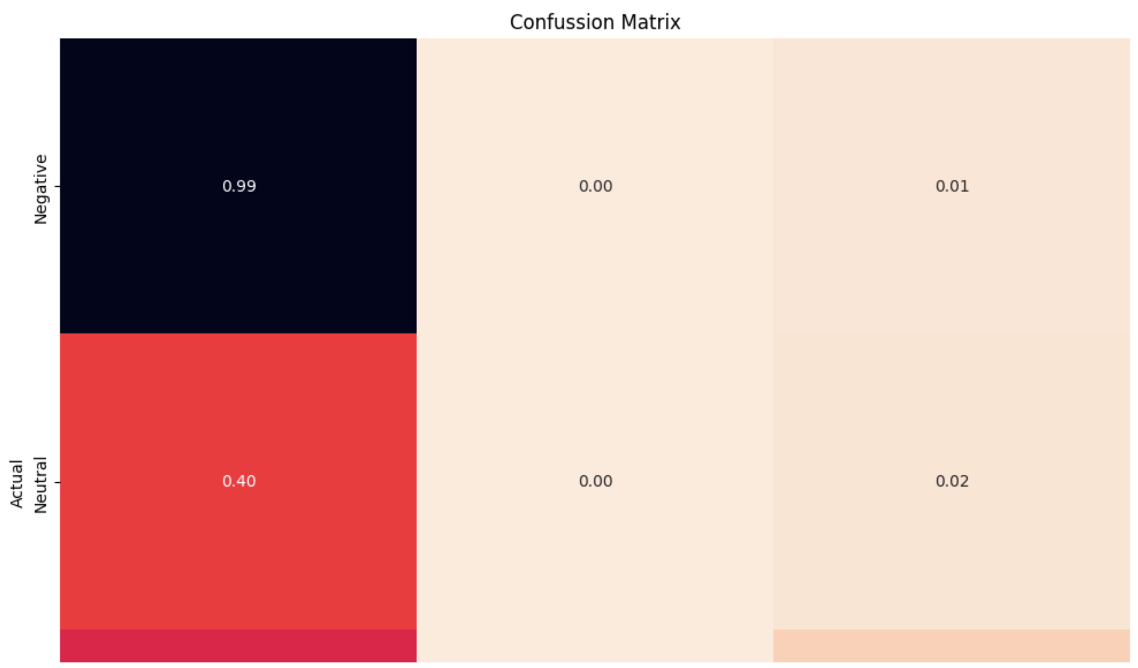
```
Testing accuracy for DT+W2V:  54.0021731256791
Testing time for DT+W2V:  0.0020236968994140625
```

Confusion Matrix

In [96]:
```
print(classification_report(y_test_tf, pred6))
```

```
              precision    recall  f1-score   support

    Negative       0.54      0.99      0.70      1451
     Neutral       0.00      0.00      0.00       597
    Positive       0.69      0.07      0.12       713

    accuracy                           0.54      2761
   macro avg       0.41      0.35      0.27      2761
weighted avg       0.46      0.54      0.40      2761


G:\Anaconda2\lib\site-packages\sklearn\metrics\_classification.py:1344: Undef
inedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
G:\Anaconda2\lib\site-packages\sklearn\metrics\_classification.py:1344: Undef
inedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
G:\Anaconda2\lib\site-packages\sklearn\metrics\_classification.py:1344: Undef
inedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [97]:
```
fig = plt.figure(figsize = (12,10)) #Define figure size
mat = confusion_matrix(y_test_tf, pred6) #Define confussion matrix with test a
mat = mat/np.sum(mat, axis = 1) #Normalizing
name = ['Negative', 'Neutral', 'Positive'] #Define names of the class accordin

#Heatmap
sns.heatmap(mat, annot = True, cmap = 'rocket_r', fmt = '.2f',xticklabels=name
plt.title('Confussion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')

plt.show()
```



In [ ]:

In [132]:
```
!pip freeze > requirements.txt
```

```
WARNING: Ignoring invalid distribution -cipy (g:\anaconda2\lib\site-packages)
```

In [ ]: