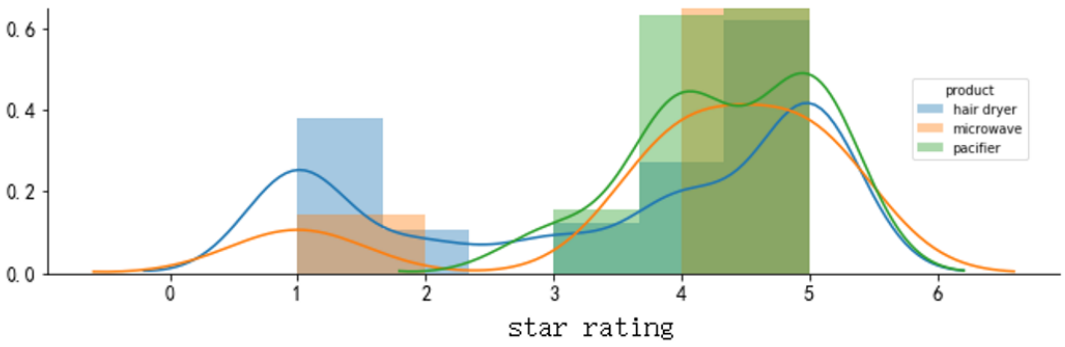
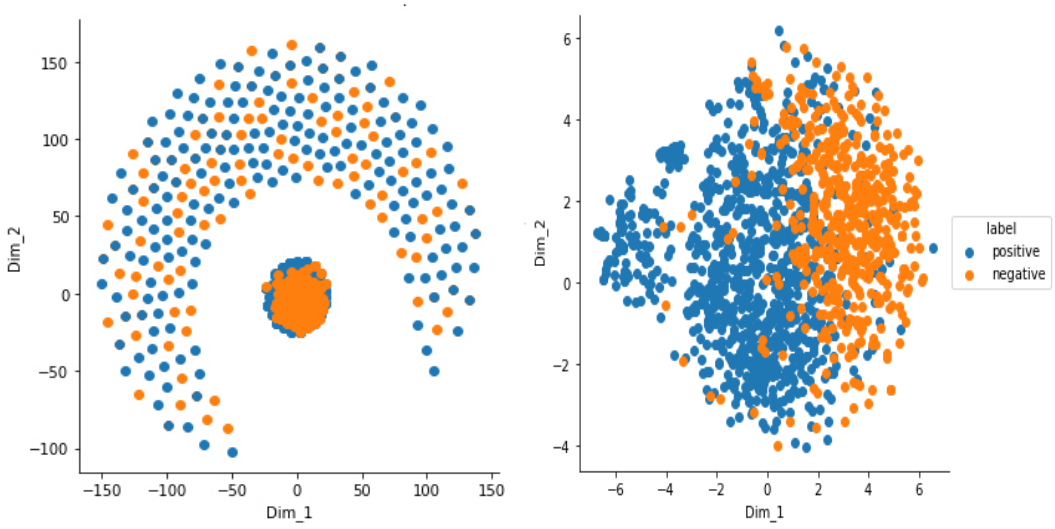
100%

x	
w	x
y	
C	
h	
v	
NPS	
P	Δt
R	
H	
A	
B	
$Supp(X \bigcap Y)$	
$Conf(X \leftarrow Y)$	
D	
Γ_i	P_i



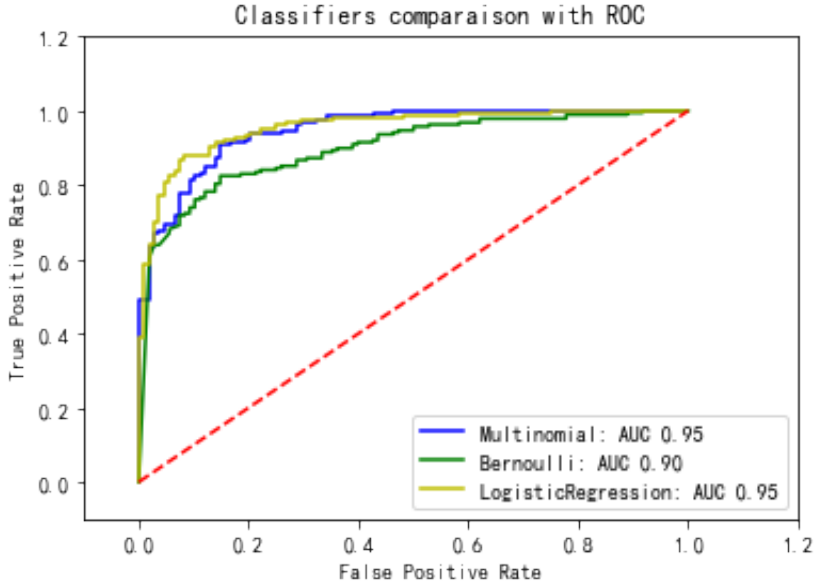
$$(X) = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right]$$

$$NPS = (Promoters - Detractors)/Totalratings * 100$$



$$y = \frac{1}{1 + e^{-(w^T x + b)}}$$

$$xyL = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$



$$|coefficient| > 1$$

RE

$$E_i = \sum_{n=1}^{N} e_n, e_n \in R$$

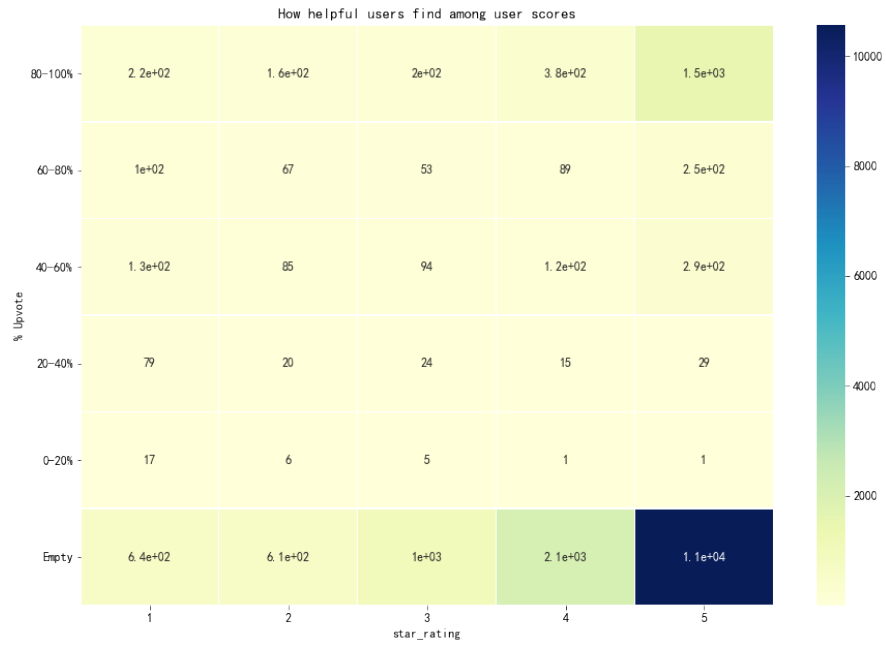
NRe_nR

...	...	...

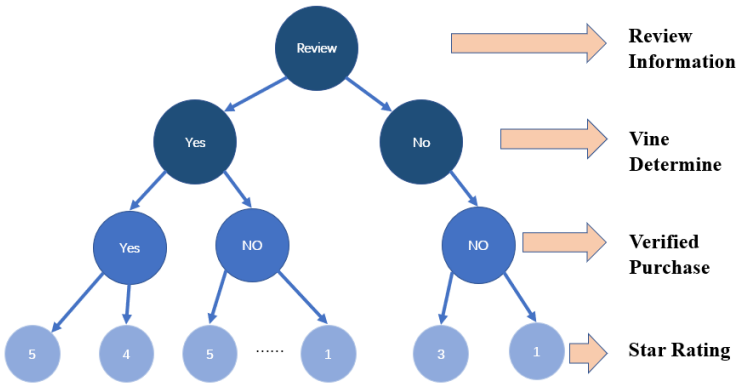
C

$$C = \frac{h}{t}(t \neq 0)$$

$htt = 0C0C$



C



C

. . . .

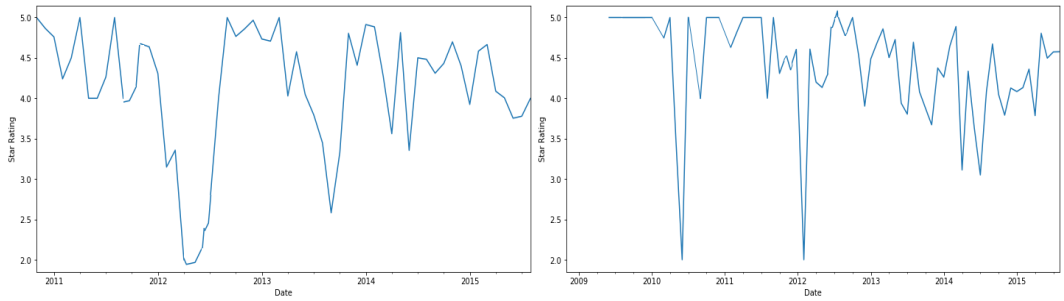
ΔT

$$P = \frac{\sum_{n=1}^{N} r \cdot H}{\sum_{n=1}^{N} H}$$

$P \Delta T N r H$

$$H = \begin{cases} C, (V = 0) \\ 10C, (V = 1) \end{cases}$$

$CVV10$



$I = \{i_1, i_2, \cdots, i_m\} iD = \{I\} k - lengthII_k X \subseteq I_x Y \subseteq I_y (x, y \in \{1, 2, \cdots, n\}) X$
Y

$$Supp(X \bigcap Y) = \frac{num(X \bigcap Y)}{num(AllSample)}$$

$num(X \bigcap Y) num(AllSample) XY X \bigcap Y Sup(X, Y) > Sup_{min} Sup_{min} XY$

$$Conf(X \leftarrow Y) = \frac{num(XY)}{num(Y)}$$

$XY Conf(X \leftarrow Y) > Conf_{min} Conf_{min}$
X

$$Supp(X) \geqslant Supp(X \bigcap Y) > Supp_{min}$$

XX

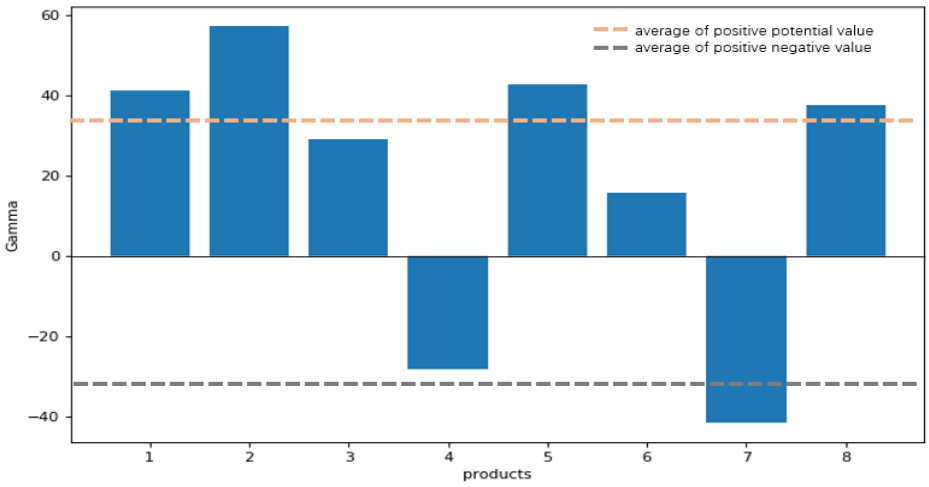
$$Supp_{min} > Supp(X) \geqslant Supp(X \bigcap Y)$$

	[]		
	[]		
	.....	...	...
	[]		
	[]		
	.....	...	...
	[]		
	[]		

$P_i\Gamma_i$

$$\Gamma_i = \frac{(\alpha_i - \beta_i)}{t_i} \times 100$$

$\alpha_i\beta_iP_it_iP_i$



$\Gamma\Gamma$

$$A = [a_1, a_2, \cdots, a_s] B = [b_1, b_2, \cdots, b_s] a_i b_i t_i t_{i+1} \{t_i\} \Delta t t_{i+1} - t_i = \Delta t AB$$

$$a_i \leftarrow a_i / a_{max}$$
$$b_i \leftarrow b_i / b_{max}$$

$$a_{max} b_{max} \{b_1, \cdots, b_m\} \, (1 \leqslant m < s) BBB = [b_1, b_2, \cdots, b_t] (t < s) a_i b_j \delta(a_i, b_j)$$

$$D = \frac{\sum_{n=1}^{N} \delta(a_i, b_j) \cdot W_n}{\sum_{n=1}^{N} W_n}$$

$DABD$

$A = [a_1, a_2, \cdots, a_s]$
$B = [b_1, b_2, \cdots, b_t]$

δ

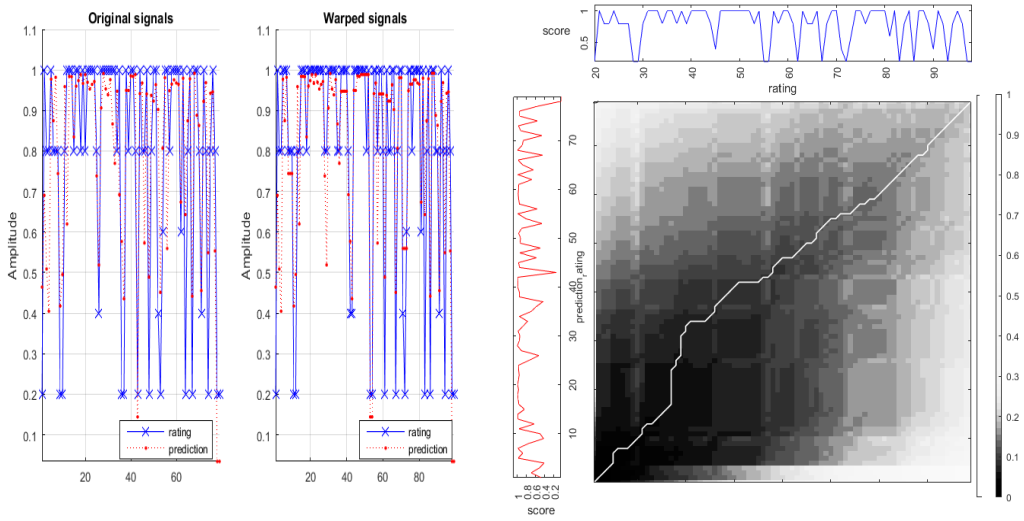
$m[1, 1, 1:2] \leftarrow (\delta(a, b), (0, 0))$

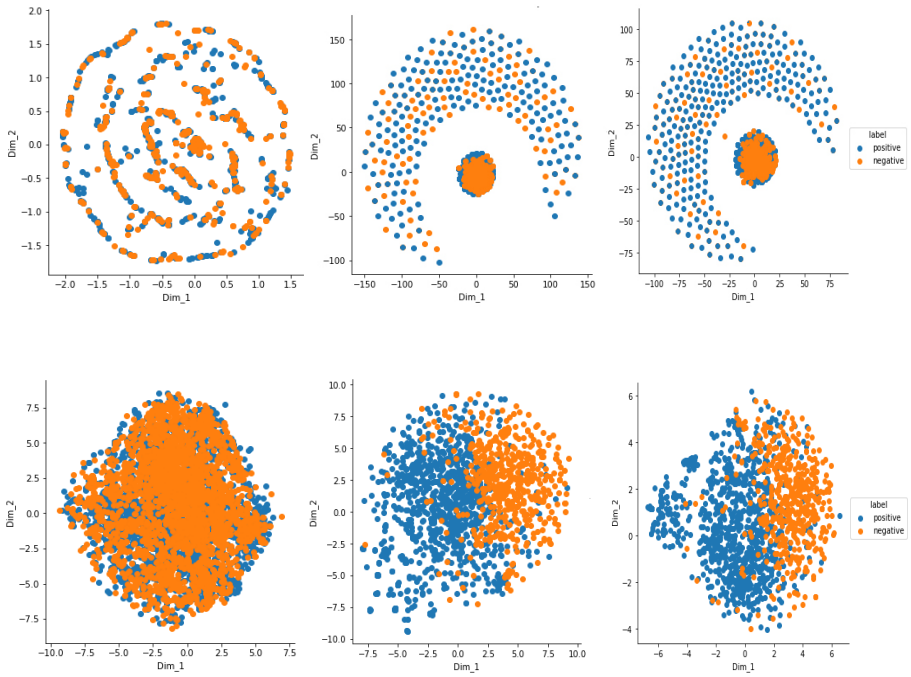
$m[i, 1, 1:2] \leftarrow (m[i, 1, 1] + \delta(a_i, b_1), (i - 1, 1))$

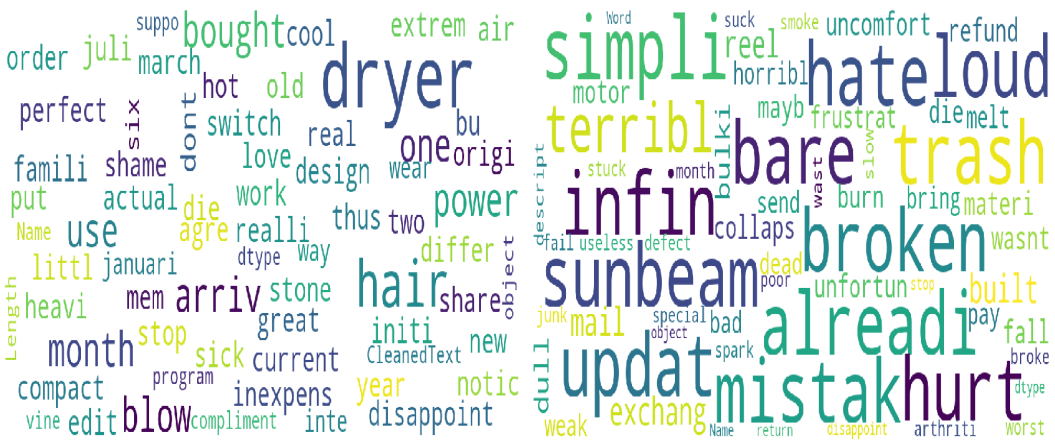
$m[1, j, 1:2] \leftarrow (m[1, j - 1, 1] + \delta(a_1, b_j), (1, j - 1))$

$minimum \leftarrow minVal(m[i - 1, j, 1], m[i, j - 1, 1], m[i - 1, j - 1, 1])$
$m[i, j, 1:2] \leftarrow (first(minimum) + \delta(a_i, b_j), second(minimum))$

$M[S, T]$

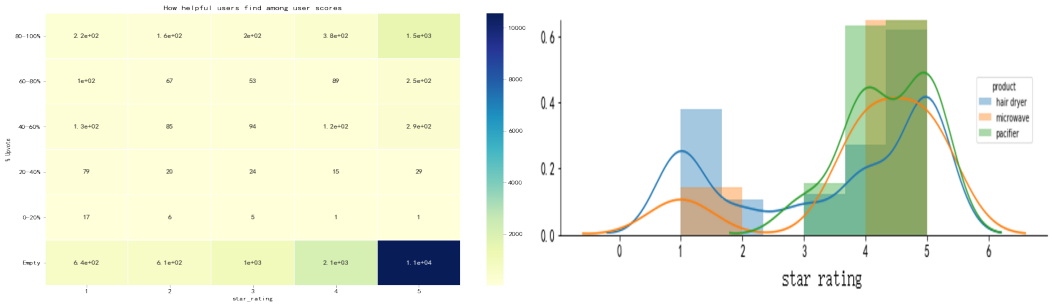

Original signals
Warped signals
Amplitude
rating
prediction
score
prediction_rating


Dim_1
Dim_2
label
positive
negative


order juli march suppo bought cool extrem air dryer hot old perfect six dont switch real bu one origi famili shame love design wear put actual die work power Name use agre realli thus two littl januari dtype Way differ Length heavi mem arriv stone hair share object great month stop sick current initi CleanedText new compact program inexpens year notic vine edit blow compliment inte disappoint
Word simpli suck smoke uncomfort refund reel horribl hateloud motor terribl mayb frustrat die melt bulki bare trash descript stuck month infin send burn bring materi wast slow fail useless defect collaps broken wasnt sunbeam dead poor unfortun stop built junk mail special object bad pay fall alreadi dull updat spark broke mistakhurt dtype weak exchang Name return disappoint arthriti worst

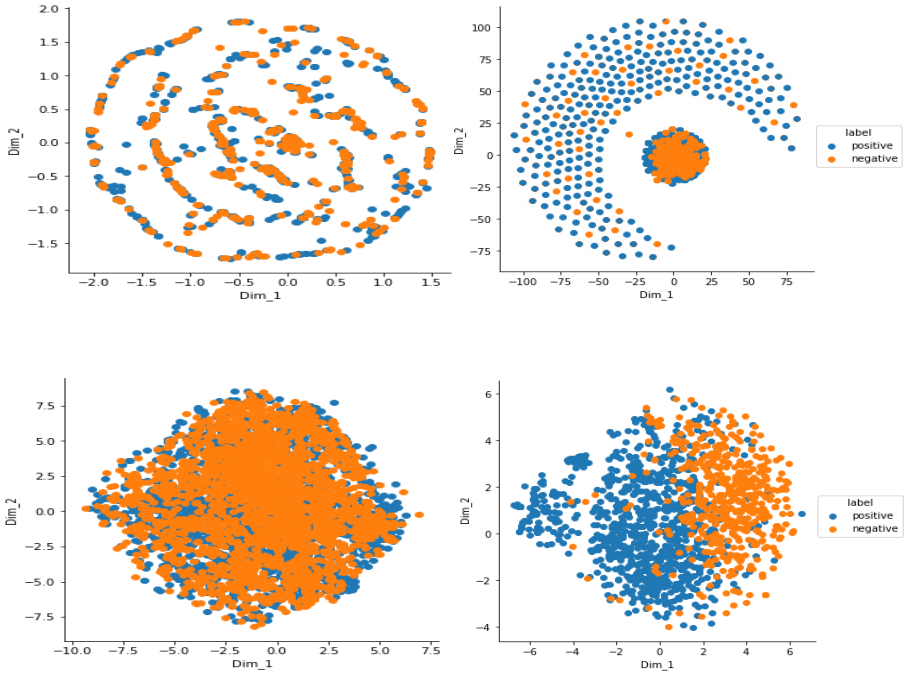
&

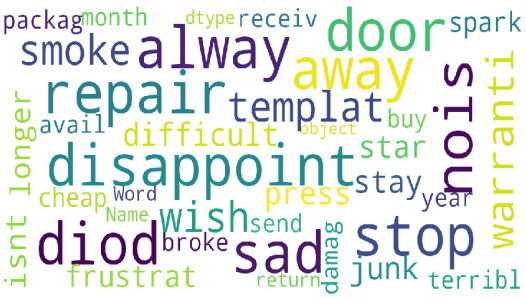
&




```
df['Helpful␣%'] = np.where(df['helpful_votes'] > 0,
                  df['helpful_votes'] / df['total_votes'], -1)
df['%␣Upvote'] = pd.cut(df['Helpful␣%'],
        bins = [-1, 0, 0.2, 0.4, 0.6, 0.8, 1.0],
        labels = ['Empty', '0-20%', '20-40%',
                  '40-60%', '60-80%', '80-100%']
                      , include_lowest = True)
df.head()
df_s = df.groupby(['star_rating', '%␣Upvote'])
                  .agg({'review_id': 'count'})
df_s = df_s.unstack()
df_s.columns = df_s.columns.get_level_values(1)
fig = plt.figure(figsize=(15,10))
sns.heatmap(df_s[df_s.columns[::-1]].T,
            cmap = 'YlGnBu', linewidths=.5, annot = True)
```

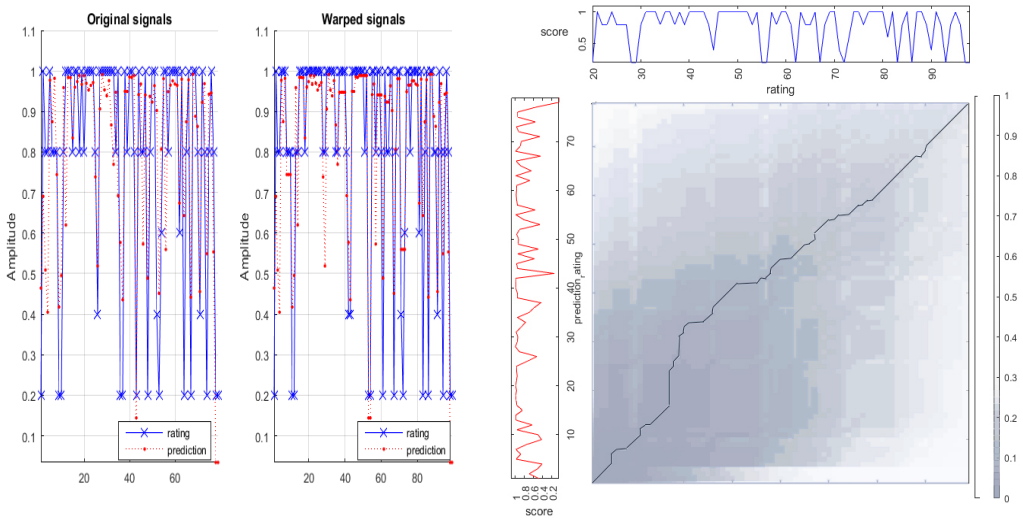
```
plt.yticks(rotation=0)
plt.title('How␣helpful␣users␣find␣among␣user␣scores')
plt.show()
```





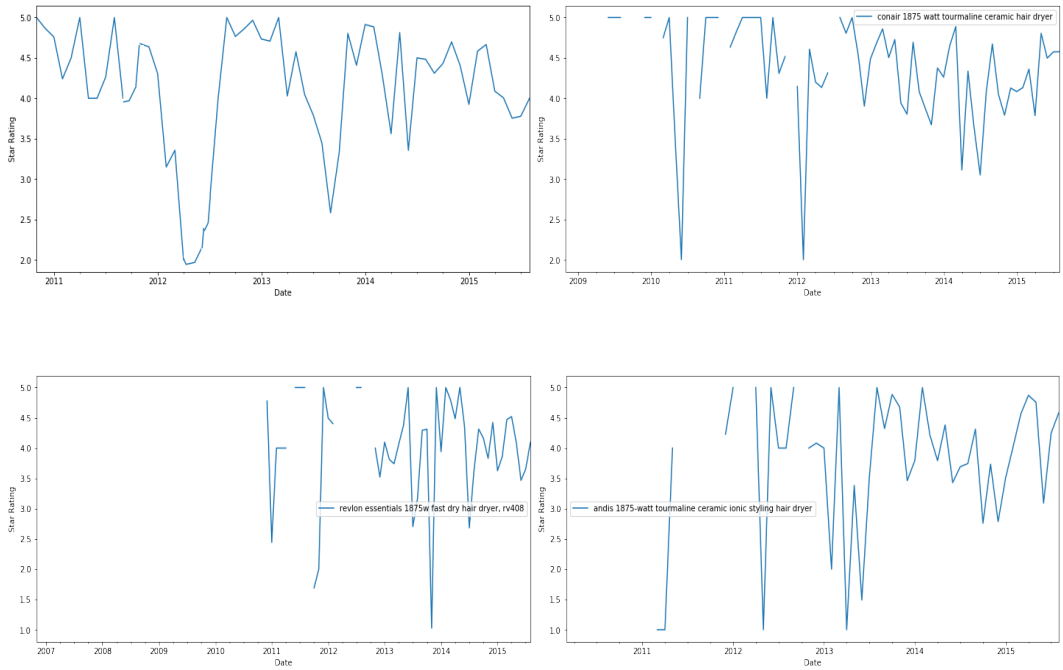

```
import re
# Important steps to clean the text data.
filtered_data = df[df['star_rating'] != 3]
def partition(x):
    if x>3:
        return 'positive'
    return 'negative'

actual_score = filtered_data['star_rating']
positiveNegative = actual_score.map(partition)
filtered_data['Score'] = positiveNegative
filtered_data.head()
```

```
# Defining function to clean html tags
def cleanhtml(sentence):
    cleaner = re.compile('<.*>')
    cleantext = re.sub(cleaner, ' ', sentence)
    return cleantext

# Defining function to remove special symbols
def cleanpunc(sentence):
    cleaned = re.sub
        (r'[?|.|!|*|@|#|\'|"|,|)|(|\|/]',r'',sentence)
    return cleaned


def text_fit(X, y, model, clf_model, coef_show=1):
    X_c = model.fit_transform(X)
    print('# features: {}'.format(X_c.shape[1]))
    X_train, X_test, y_train, y_test = \
        train_test_split(X_c, y, random_state=0)
    print('# train records: {}'.format(X_train.shape[0]))
    print('# test records: {}'.format(X_test.shape[0]))
    clf = clf_model.fit(X_train, y_train)
    acc = clf.score(X_test, y_test)
    print('Model Accuracy: {}'.format(acc))

    if coef_show == 1:
        w = model.get_feature_names()
        coef = clf.coef_.tolist()[0]
        coeff_df = pd.DataFrame({'Word': w, 'Coefficient': coef})
        coeff_df = coeff_df.sort_values\
            (['Coefficient', 'Word'], ascending=[0, 1])
        print('-Top 20 positive-')
        print(coeff_df.head(20).to_string(index=False))
        print('')
        print('-Top 20 negative-')
        print(coeff_df.tail(20).to_string(index=False))
    return coeff_df


coeff_df = text_fit(X, y, c, LogisticRegression())
```




```
function [Dist,D,k,w,rw,tw]=dtw(r,t,pflag)
[row,M]=size(r); if (row > M) M=row; r=r'; end;
[row,N]=size(t); if (row > N) N=row; t=t'; end;
d=sqrt((repmat(r',1,N)-repmat(t,M,1)).^2);
 %this makes clear the above instruction Thanks Pau Mic
D=zeros(size(d));
D(1,1)=d(1,1);
for m=2:M
    D(m,1)=d(m,1)+D(m-1,1);
end
for n=2:N
    D(1,n)=d(1,n)+D(1,n-1);
end
for m=2:M
    for n=2:N
        D(m,n)=d(m,n)+min(D(m-1,n),min(D(m-1,n-1),D(m,n-1)));
    end
end
Dist=D(M,N);n=N;m=M;k=1;w=[M N];
while ((n+m)~=2)
    if (n-1)==0
        m=m-1;
    elseif (m-1)==0
        n=n-1;
    else
      [values,number]=min([D(m-1,n),D(m,n-1),D(m-1,n-1)]);
```

```
      switch number
      case 1
        m=m-1;
      case 2
        n=n-1;
      case 3
        m=m-1;n=n-1;
      end
  end
    k=k+1;
    w=[m n; w]; % this replace the above sentence.
end
% warped waves
rw=r(w(:,1));
tw=t(w(:,2));
end
```




```
rating_series = pd.DataFrame(kindle.review_date)
dforms=[]
for x in rating_series.review_date:
    dforms.append((pd.to_datetime(x)).value)
# now we have dforms which has dates transformed to numeric values
rating2 = rating_series.assign(date_min = dforms)
rating2.reset_index(inplace=True)
#rating2.set_index('date_min')
#rating2.columns=['timestamp_string','review_count','date_min']
bins = np.linspace(min(rating2.date_min),max(rating2.date_min),num
rating2.hist(column='date_min', bins=20,figsize=(10,6),)
rating2.hist(column='date_min', bins=30,figsize=(10,6))
rating2.hist(column='date_min', bins=50,figsize=(10,6))

def NPS_eval (A):
    score =0
    for x in A[:]:
        if (x>4) :
            score+=1
        elif (x<3) :
            score-=1
    return 100*score/len(A)
```

```
NPS_overtime = kindle[['temp','star_rating']]
NPS_overtime.groupby(by='temp').agg(NPS_eval).plot(figsize=(15,10)

for i in range(8):
    title = final['product_title'].value_counts().index[i]
    XXXX = final[final['product_title']==title]
    month = XXXX.resample('M').sum()
    month['H/P'] = month['H']/month['P']
    month_dates = month['H/P']
    month_dates.sort_index(inplace=True)
    month_dates.plot(figsize=(12,6))
    plt.legend([title])
    plt.ylabel('Star␣Rating')
    plt.show()
```

...	...	...	...	...

D
Sup_{min}
$Conf_{min}$
F

$t \leftarrow 1$
$C_t = \varnothing$
$length = 1$

I_i
I_i

$I_i(j) \notin C_t$
$C_t = C_t \cup I_i(j)$

$F_t = \{f | f \in C_t, Sup(f) > Sup_{min}\}$
$F \neq \varnothing$

$C_t \leftarrow F_{t-1}$
$F_t = \{f | f \in C_t, (Sup(f) > Sup_{min}) \bigcap (Comf(f) > Conf_{min})\}$

F_{t-1}

		[]		
		[]		
		.....	...	...
		[]		
		[]		
		.....	...	...
		[]		
		[]		