Methodology Document

For Cell Phone Customer Review Analysis

Agenda

- Data import and pre-processing
- Exploratory Data Analysis
- Text Analytics
- Machine Learning
- Tableau Visualization

Data Import & Pre-Processing

Import Data

- Import json file containing meta data and convert to Pandas Data Frame
- Import csv file containing reviews and other columns and convert to Data Frame
- Pre-process unix date to date & time
- Merge both data frames on asin

In [12]:
```
#merging both data frames to one
result = pd.merge(df_phonedata,df_meta,on='asin')
```

In [13]:
```
result.head(2)
```

Out[13]:

	overall	verified	reviewerID	asin	style	reviewerName	reviewText	summary	vote	image_x	...	tech2	brand	feature	rar
0	5.0	True	A24E3SXTC62LJI	7508492919	{'Color:': ' Bling'}	Claudia Valdivia	Looks even better in person. Be careful to not drop your phone so often because the rhinestones will fall off (duh). More of a decorative case than it is protective, but I will	Can't stop won't stop looking at it	NaN	NaN	...		ELEGANI	[Fashionable with unique 3D butterfly design for the iPhone 5 from others, Precise cut design with access to all ports and features of the iPhone 5, Dirt-resistant Butterfly Case for iPhone 5 with 3D PC	[>#228,5 in C Phones Accessori (See Top 1(in C Phones Accessorie >#114,920 Cell Phon

Check Null Values

- Drop columns with <45% null values
- Drop records where columns contain less that 1% null values
- Drop redundant columns which are not required for analysis
- Price columns contains empty and redundant values, need to be fixed

```
def nullval(df):
    return round((df.isnull().sum()*100/len(df)).sort_values(ascending = False),2)
```

```
null_column_45 = nullval(result)[nullval(result)>45]
```

Price Null Values

- Replace non price text with np.nan
- Fill null values with median price

```
cleaned_price = []
for i in filtered_3.price:
    if type(i)==str and not ('-' in i):
        y = i.replace(',', '')
        x = float(y.strip('$'))
        cleaned_price.append(x)

    elif type(i)==str and ('-' in i):
        x = i.split(' - ')
        y = [float(x[0].strip('$')),float(x[1].strip('$'))]
        z = (y[0]+y[1])/2
        cleaned_price.append(z)

    else:
        cleaned_price.append(i)
```

```
[58]: x = [filtered_3['cleaned_dollar_price']]
      for i in x:
          print('mean =',i.mean(),'\n',
          'median =',i.median(),'\n',
              'mode =',i.mode(),'\n')

      mean = 19.149420117853914
       median = 9.99
       mode = 0    7.99
      dtype: float64
```

```
[59]: filtered_3['cleaned_dollar_price'].fillna(9.99, inplace = True)
```

Text Pre-Process and Data Filter

```
def preprocess(ReviewText):
    ReviewText = ReviewText.str.replace("(<br/>)", "")
    ReviewText = ReviewText.str.replace('(<a).*(>).*(</a>)', '')
    ReviewText = ReviewText.str.replace('(&amp)', '')
    ReviewText = ReviewText.str.replace('(&gt)', '')
    ReviewText = ReviewText.str.replace('(&lt)', '')
    ReviewText = ReviewText.str.replace('(\xa0)', ' ')
    return ReviewText
```

```
# checking latest date
max(df2['ReviewDateTime'])
```

```
Timestamp('2018-10-02 00:00:00')
```

```
# Filter data between two dates
filtered_1 = df2.loc[(df2['ReviewDateTime'] >= '2015-10-02')
                     & (df2['ReviewDateTime'] < '2018-10-02')]
```

```
# creating a new dataframe to contain only relevant columns for further analysis
filtered_2 = filtered_1[['asin','ReviewDateTime',
                         'overall','reviewText',
                         'word_count','review_sentiment',
                         'main_cat''feature_str',
                         'word_count_features',
                         'brand','price']]
```

```
# creating dataframe conating only smartphone related data
filtered_3 =filtered_2[filtered_2['main_cat'].str.contains('Cell Phones & Accessories')==True]
```

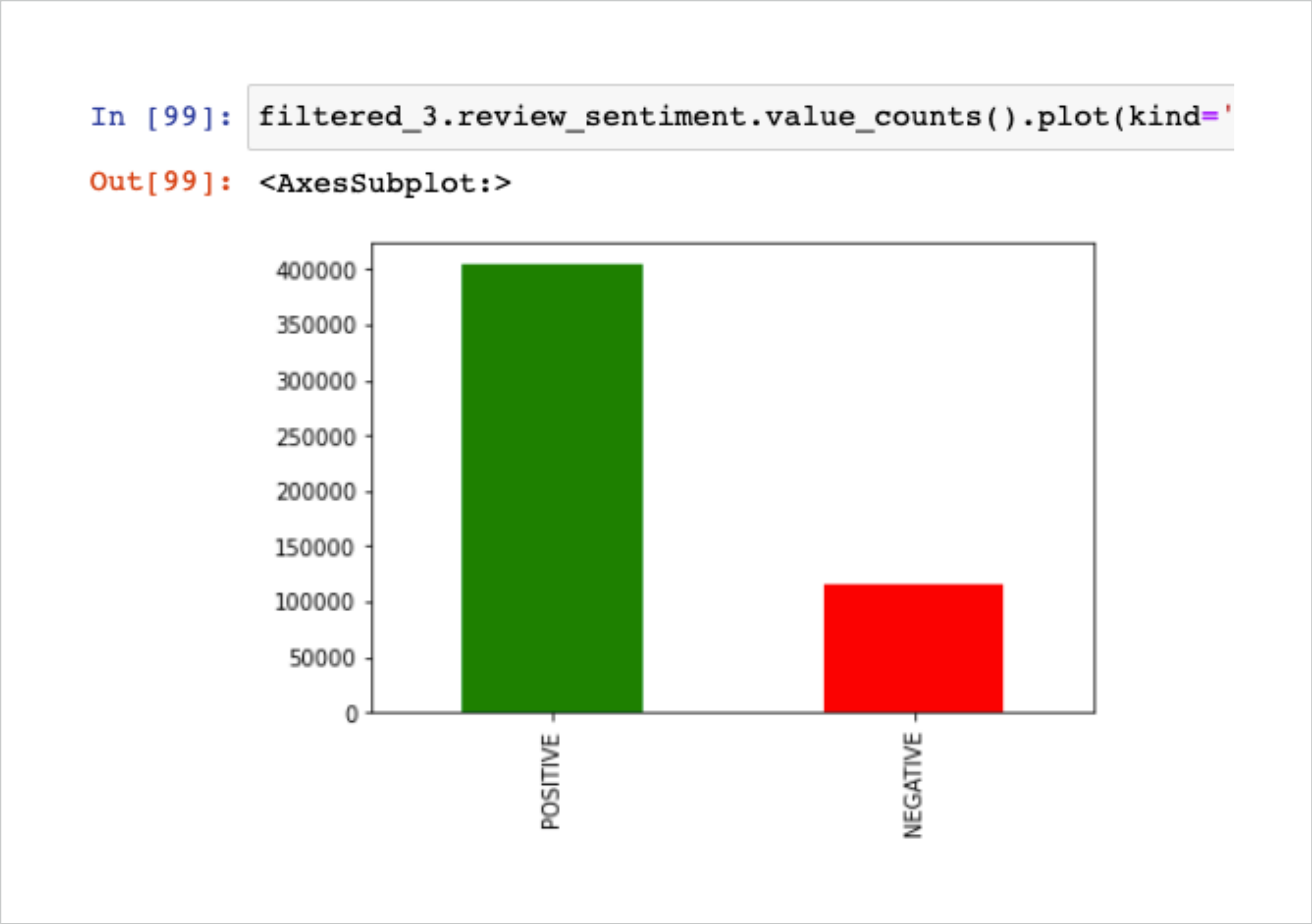
- Create a function to clean text
- From a business standpoint, it is eminent to consider the fact that in the ever-changing tech industry we need to gain insights into the latest market trends.
- Thus, only further analyze cell phone related data of the last 3 years

Data Pre-processing Summary

- Joined reviews and product information on ASIN
- Treated null values, redundant columns and data discrepancies
- Created new columns containing feature and review word count
- Filtered data to keep only records containing cell phone related data from the last 3 years

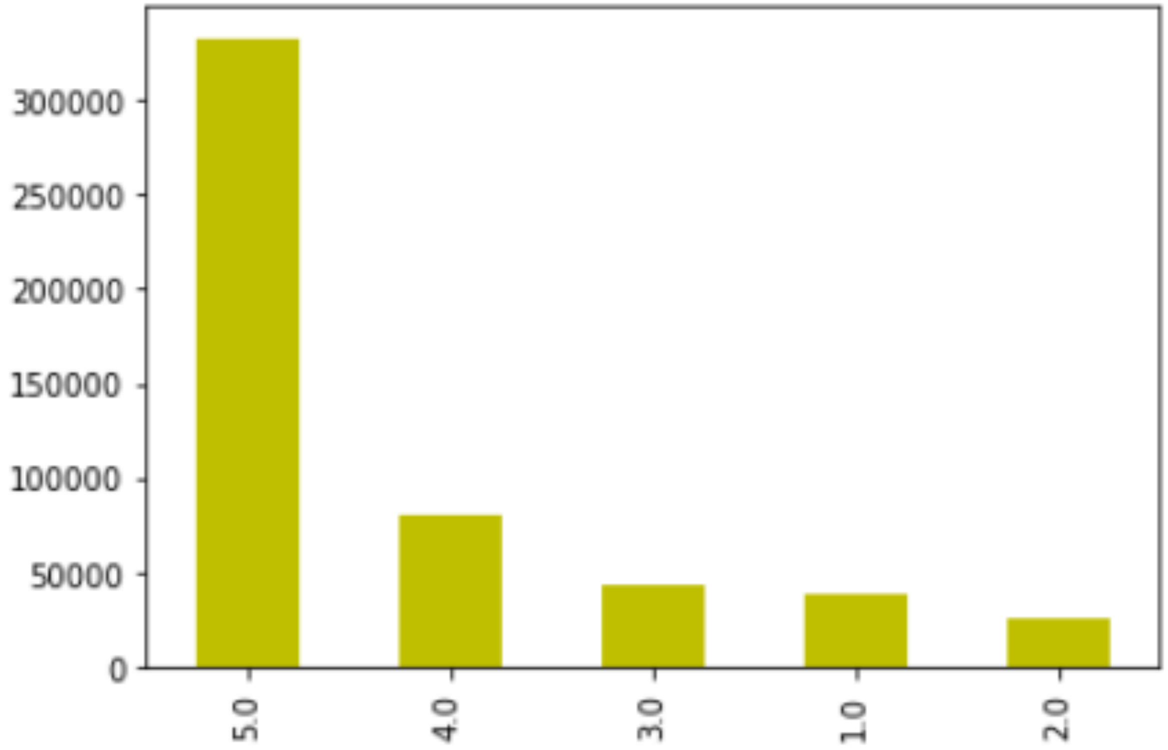
Exploratory Data Analysis

Sentiment Analysis- Bar Graph



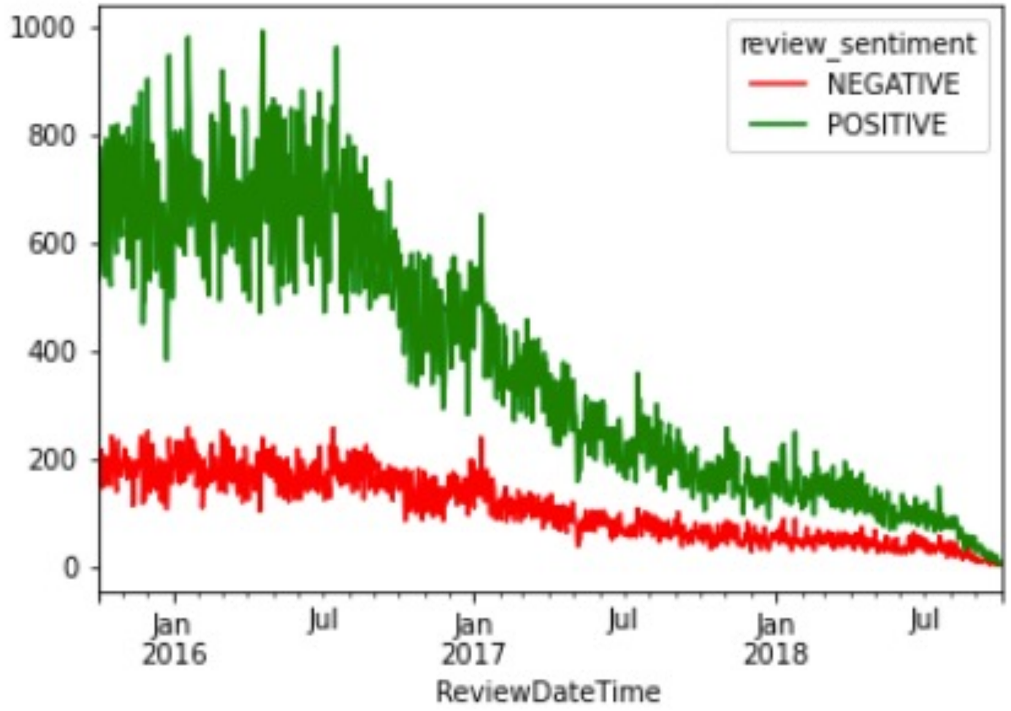
Overall Rating

```
In [100]: filtered_3.overall.value_counts().plot(kind='bar'

Out[100]: <AxesSubplot:>
```



Sentiment Trend

```
In [104]: group_time_sent = filtered_3.groupby(by='ReviewDateTime'
          unstacked = group_time_sent.unstack(level=1)
          unstacked.plot.line(color=['r','g'])

Out[104]: <AxesSubplot:xlabel='ReviewDateTime'>
```



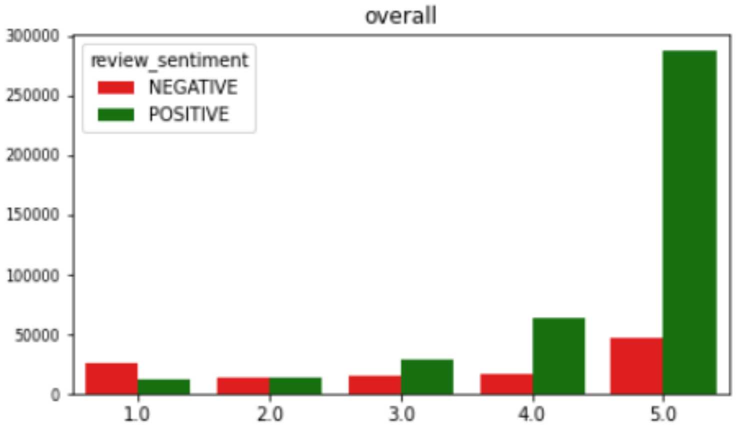
Overall Sentiment Split

```python
group_brand_sent = filtered_3.groupby(by='brand')['review_sentiment'].value_counts()
unstacked1 = group_brand_sent.unstack(level=1)
```

```python
unstacked1 = group_brand_sent.unstack(level=1)
```

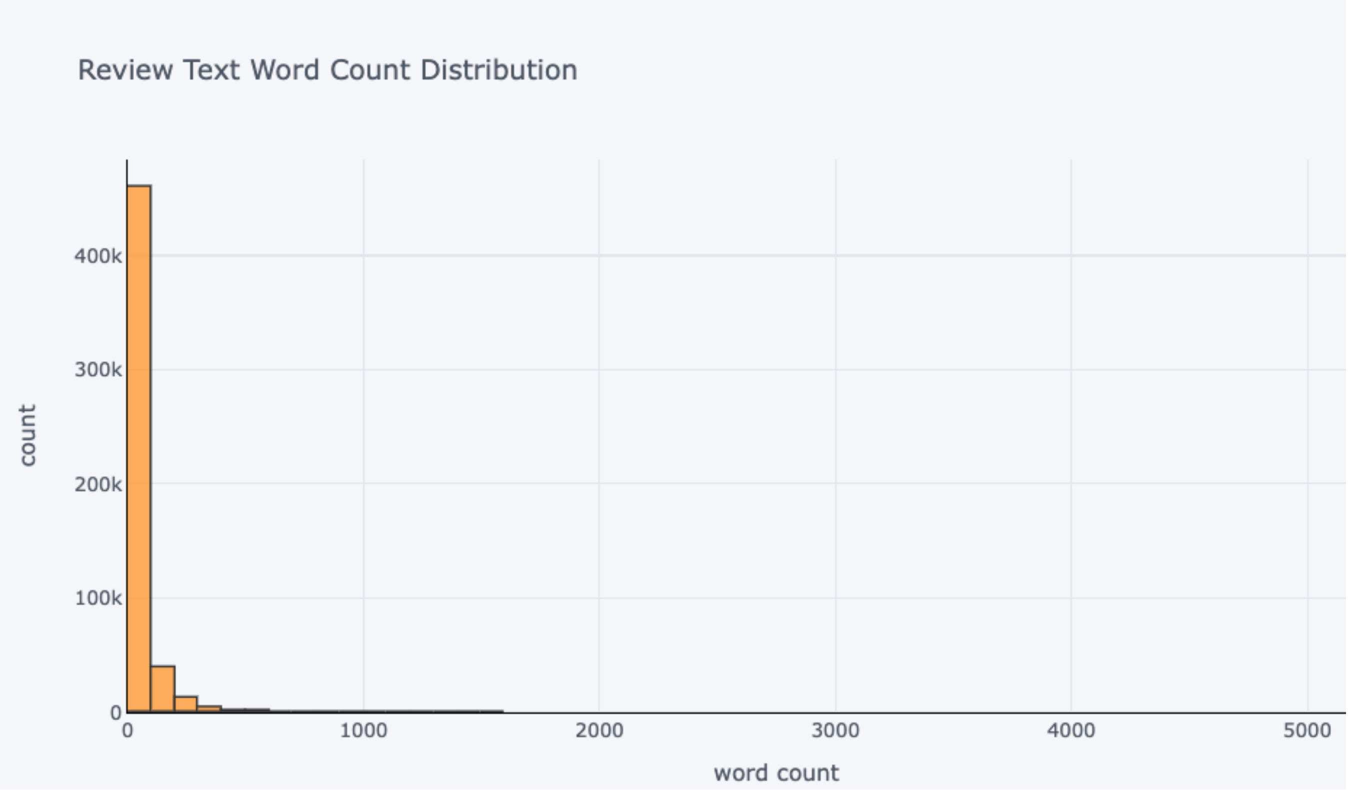
```python
# Plotting all the graph to find the relation and evaluting for dropping such columns

plt.figure(figsize = [30,30])
plt.subplot(7,4,2)
ax = sns.countplot(filtered_3['overall'], hue = filtered_3["review_sentiment"], palette = ["r","
plt.yticks(fontsize=8)
plt.xlabel("")
plt.ylabel("")
plt.title('overall')
```

```
Text(0.5, 1.0, 'overall')
```



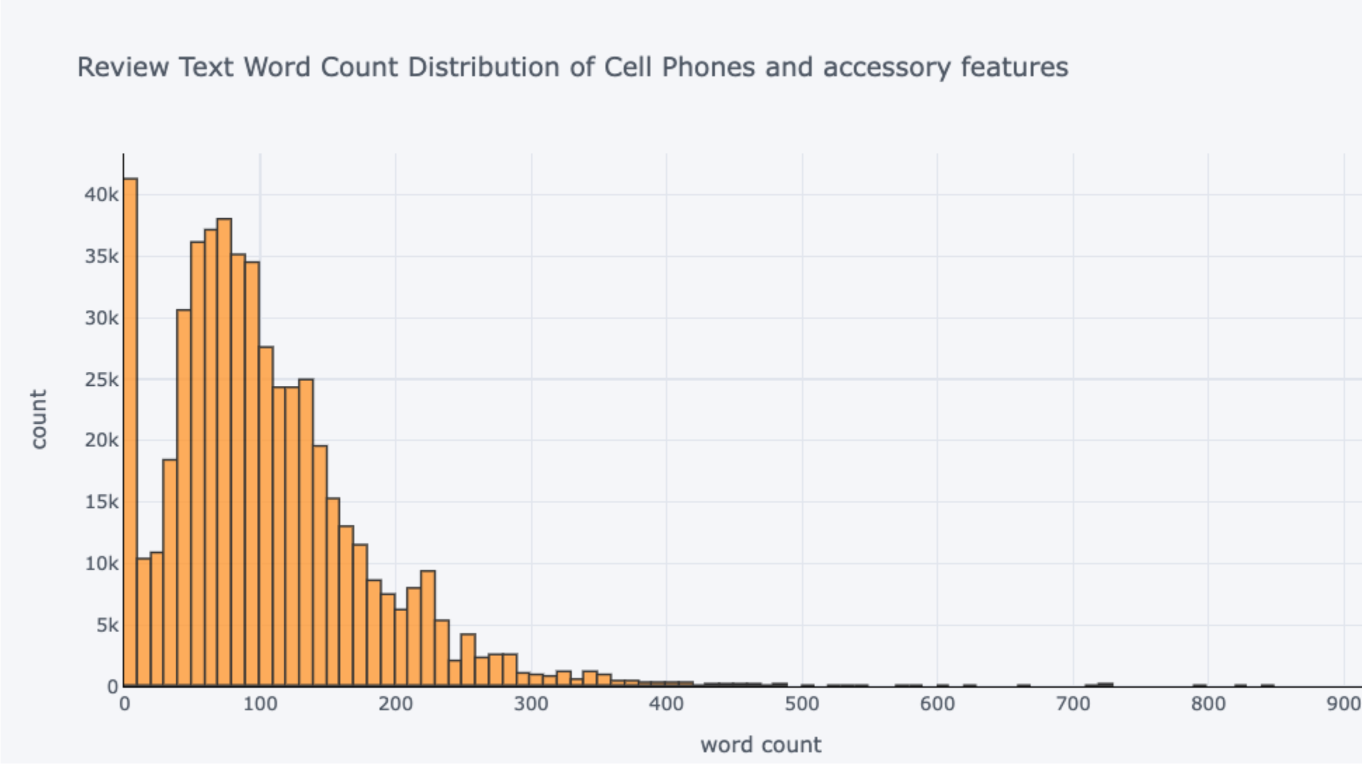
Review Word Count

```
In [102]: filtered_3['word_count'].iplot(
    kind='hist',
    bins=100,
    xTitle='word count',
    linecolor='black',
    yTitle='count',
    title='Review Text Word Count Distribution')
```



Feature Word Count

```
In [111]: # cell phone feature word count
filtered_3['word_count_features'].iplot(
    kind='hist',
    bins=100,
    xTitle='word count',
    linecolor='black',
    yTitle='count',
    title='Review Text Word Count Distribution of Cell Phones and accessory features')
```



Export “dataset” as csv to visualize in Tableau

```python
# compile documents
dataset = filtered_3[['asin',
 'ReviewDateTime',
 'overall',
 'reviewText',
 'word_count',
 'review_sentiment',
 'main_cat',
 'feature_str',
 'word_count_features',
 'brand',
 'cleaned_dollar_price']]
```

```python
# Add cleaned docs to the dataset
dataset['doc_clean'] = doc_clean
```

```python
dataset['feature_clean'] = doc_clean_feature
```

```python
dataset.head(2)
```

	asin	ReviewDateTime	overall	reviewText	word_count	review_sentiment	main_cat	feature_str	word_count_features	brand	cleaned_dollar_price	doc_clean	feature_clean
77	7532385086	2017-03-30	1.0	I didn't like this	4	NEGATIVE	Cell Phones & Accessories	['Rubberized Purple Wave Flower Snap on Design Case Hard Case Skin Cover Faceplate for Sprint Htc Evo 4g']	18	Generic	9.99	[didnt, like]	[rubberized, purple, wave, flower, snap, design, hard, skin, cover, faceplate, sprint, htc, evo, 4g]
78	7532385086	2015-11-08	2.0	It didn't fit my phone.	5	POSITIVE	Cell Phones & Accessories	['Rubberized Purple Wave Flower Snap on Design Case Hard Case Skin Cover Faceplate for Sprint Htc Evo 4g']	18	Generic	9.99	[didnt, fit, phone]	[rubberized, purple, wave, flower, snap, design, hard, skin, cover, faceplate, sprint, htc, evo, 4g]

```python
dataset.shape
```

```
(519647, 13)
```

```python
# exporting this cleaned and pre proccessed data set
dataset.to_csv('/Users/raphael/Desktop/Capstone Data/dataset.csv')
```

Exploratory data analysis Summary

- Plotted various graphs to analyse distribution of data
- Conduct sentiment analysis of on the spread of data
- Exported the cleaned and filtered data set as a csv to analyze in Tableau
- Initial analysis indicate that the top 5 brands are 'Samsung’, 'Motorola’, ‘BLU’, 'LG' and 'Apple'

Text Analytics

Spacy for Stop words

```
In [65]: # Reading stop words from a text file in to a list
         stop_words = [line.rstrip('\n') for line in open('/Users/raphael/Desktop/Capstone Data/stop_words_long.txt')]

In [66]: # import modules required
         import nltk
         from nltk.tokenize import sent_tokenize
         from nltk.metrics.distance import jaccard_distance
         from nltk.corpus import wordnet
         from nltk.corpus import stopwords
         from nltk.stem.wordnet import WordNetLemmatizer
         from nltk import ngrams
         from wordcloud import WordCloud
         from collections import Counter
         from PIL import Image
         from wordcloud import ImageColorGenerator

         import string
         import spacy
         from spacy.lang.en.stop_words import STOP_WORDS

In [67]: # create stop word list containg spacy stop words as well as stop words data provided
         nlp = spacy.load('en_core_web_sm')
         stopwords = list(STOP_WORDS)+stop_words
```

Clean Text
Punctuations
Lemmatization

```python
# cleaning and lemmatizing for textual data
stop = (set(stopwords))
exclude = set(string.punctuation)
lemma = WordNetLemmatizer()
def clean(doc):
    stop_free = " ".join([str(i) for i in doc.lower().split() if i not in stop])
    punc_free = ''.join(ch for ch in stop_free if ch not in exclude)
    normalized = " ".join(lemma.lemmatize(word) for word in punc_free.split())
    return normalized
```

Feature and Reviews

2) Creating a dataframe contaning reviews/ features words and its word counts

```
In [80]: # creating file to export word count to create positive and negative wordclouds in tableau

# positive
pos_df = dataset[dataset.review_sentiment.isin(['POSITIVE'])]

pos_key=[]
for i in pos_df['doc_clean']:
    for j in i:
        pos_key.append(j)

pos = dict(Counter(pos_key))
pos_l = list(pos.items())


# negative
neg_df = dataset[dataset.review_sentiment.isin(['NEGATIVE'])]

neg_key=[]
for i in neg_df['doc_clean']:
    for j in i:
        neg_key.append(j)

neg = dict(Counter(neg_key))
neg_l = list(neg.items())


# creating file to export word count to create positive and negative wordclouds in tableau

# positive
pos_df3 = dataset[dataset.review_sentiment.isin(['POSITIVE'])]

pos_key3=[]
for i in pos_df3['feature_clean']:
    for j in i:
        pos_key3.append(j)

pos3 = dict(Counter(pos_key3))
pos_3 = list(pos3.items())


# negative
neg_df3 = dataset[dataset.review_sentiment.isin(['NEGATIVE'])]

neg_key3=[]
for i in neg_df3['feature_clean']:
    for j in i:
        neg_key3.append(j)

neg3 = dict(Counter(neg_key3))
neg_3 = list(neg3.items())


# create dataframe and export the data

df_neg_feat = pd.DataFrame(neg_3,columns = ['Neg_feature','Neg_Count_feat'])
df_pos_feat = pd.DataFrame(pos_3,columns = ['Pos_feature','Pos_Count_feat'])

df_neg = pd.DataFrame(neg_l,columns = ['Neg_Word','Neg_Count'])
df_pos = pd.DataFrame(pos_l,columns = ['Pos_Word','Pos_Count'])
```

Feature and Reviews of Competitor Brands

2.1) Text analytics for competitor brand data

In [85]:
```
# Filter Rows by list of values
df_brand = dataset.query("brand in ('Samsung','Motorola','Apple')")
```

In [92]:
```
# creating file to export word count to create positive and negative wordclouds in tableau

# positive
pos_df4 = df_brand[df_brand.review_sentiment.isin(['POSITIVE'])]

pos_key4=[]
for i in pos_df4['doc_clean']:
    for j in i:
        pos_key4.append(j)

pos4 = dict(Counter(pos_key4))
pos_4 = list(pos4.items())


# negative
neg_df4 = df_brand[df_brand.review_sentiment.isin(['NEGATIVE'])]

neg_key4=[]
for i in neg_df4['doc_clean']:
    for j in i:
        neg_key4.append(j)

neg4 = dict(Counter(neg_key4))
neg_4 = list(neg4.items())

# feature
# positive
pos_df5 = df_brand[df_brand.review_sentiment.isin(['POSITIVE'])]

pos_key5=[]
for i in pos_df5['feature_clean']:
    for j in i:
        pos_key5.append(j)

pos5 = dict(Counter(pos_key5))
pos_5 = list(pos5.items())


# negative
neg_df5 = df_brand[df_brand.review_sentiment.isin(['NEGATIVE'])]

neg_key5=[]
for i in neg_df5['feature_clean']:
    for j in i:
        neg_key5.append(j)

neg5 = dict(Counter(neg_key5))
neg_5 = list(neg5.items())


# create dataframe and export the data

df_neg_feat_brand = pd.DataFrame(neg_4,columns = ['Neg_feature_brand','Neg_Count_feat_brand'])
df_pos_feat_brand = pd.DataFrame(pos_4,columns = ['Pos_feature_brand','Pos_Count_feat_brand'])

df_neg_brand = pd.DataFrame(neg_5,columns = ['Neg_Word_brand','Neg_Count_brand'])
df_pos_brand = pd.DataFrame(pos_5,columns = ['Pos_Word_brand','Pos_Count_brand'])
```

Export "key_df" as csv to visualize in Tableau

```
[97]: # creating a combined dataframe all words and it's respective word count
key_df = pd.concat([df_neg, df_pos,df_neg_feat,df_pos_feat,df_neg_feat_brand,df_pos_feat_brand,df_neg_brand,df_pos_brand], axis=1)
key_df.head()
```

[97]:

	Neg_Word	Neg_Count	Pos_Word	Pos_Count	Neg_feature	Neg_Count_feat	Pos_feature	Pos_Count_feat	Neg_feature_brand	Neg_Count_feat_brand	Po
0	didnt	5542.0	didnt	17356	rubberized	2608.0	rubberized	10126.0	work	1851.0	
1	like	18568.0	fit	78450	purple	155.0	purple	631.0	well	297.0	
2	worked	3447.0	phone	264771	wave	52.0	wave	210.0	idk	8.0	
3	work	20524.0	good	104870	flower	125.0	flower	463.0	looking	79.0	
4	well	6088.0	charger	44226	snap	2945.0	snap	11494.0	for	46.0	

```
[98]: # exporting this data to csv for analysis in tableau
key_df.to_csv('/Users/raphael/Desktop/Capstone Data/keyword_dataset.csv')
```

Text analytics Summary

- Text analytics for competitor brand data
- Creating a data frame containing reviews/ features words and its word counts
- Sentiment analysis on reviews and features
- Creating word clouds
- Exporting word and word count based on setiment

Machine Learning

Create Labels for Sentiments and Vectorize Bag of words

```
In [280]: # Transform sentiment to labels
          lb = preprocessing.LabelBinarizer()
          target_labels=lb.fit_transform(dataset['review_sentiment'])

In [281]: # Add transformed lables to dataset
          dataset['labels_1']=target_labels

In [282]: # Vecotorize bag of words
          tokens_raw=[" ".join(t) for t in dataset['doc_clean']]
          vectorizer = TfidfVectorizer()
          X =vectorizer.fit_transform(tokens_raw)
```

Split Data and Train Logistic Regression Model

In [283]:
```
# Features and Labels
ylabels = dataset['labels_1']

# Split data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, ylabels, test_size=0.3, random_state=42)
```

In [284]:
```
# Train Logistic Regression Model
model = LogisticRegression()
clf=model.fit(X_train,y_train)

# check score of the trained model
score =clf.score(X_train,y_train)*100
print(score)
```

```
94.77391189601707
```

Test and Predict using Models

```
In [285]: # Start predictions with X_test
          y_pred =clf.predict(X_test)

In [286]: # Check accuracy of the prediction
          metrics.accuracy_score(y_test, y_pred)

Out[286]: 0.9410244074537348
```

Multinomial

MultinomialNB

```
[105]: from sklearn.naive_bayes import MultinomialNB
```

```
[106]: # Train Multinomial Model
MNB = MultinomialNB()
clf_mnb = MNB.fit(X_train,y_train)

# check score of the trained model
score_clf_mnb =clf_mnb.score(X_train,y_train)*100
print(score_clf_mnb)
```

```
77.81671858774662
```

```
[107]: # Start predictions with X_test
y_pred_mnb =clf_mnb.predict(X_test)
```

```
[108]: # Check accuracy of the prediction
metrics.accuracy_score(y_test, y_pred_mnb)
```

```
[108]: 0.760823493793521
```

Complement

ComplementNB

```
[109]: from sklearn.naive_bayes import ComplementNB
```

```
[110]: # Train ComplementNB Model
       CNB = ComplementNB()
       clf_cnb = CNB.fit(X_train,y_train)

       # check score of the trained model
       score_clf_cnb =clf_cnb.score(X_train,y_train)*100
       print(score_clf_cnb)

       92.1858774662513
```

```
[111]: # Start predictions with X_test
       y_pred_cnb =clf_cnb.predict(X_test)
```

```
[112]: # Check accuracy of the prediction
       metrics.accuracy_score(y_test, y_pred_cnb)

[112]: 0.8658795034816833
```

Bernoulli

BernoulliNB

[113]:
```python
from sklearn.naive_bayes import BernoulliNB
```

[114]:
```python
# Train BernoulliNB Model
BNB = BernoulliNB()
clf_bnb = BNB.fit(X_train,y_train)

# check score of the trained model
score_clf_bnb =clf_bnb.score(X_train,y_train)*100
print(score_clf_bnb)
```

```
86.55244029075804
```

[115]:
```python
# Start predictions with X_test
y_pred_bnb =clf_bnb.predict(X_test)
```

[116]:
```python
# Check accuracy of the prediction
metrics.accuracy_score(y_test, y_pred_bnb)
```

[116]:
```
0.8180442022403875
```

Model Evaluation: Confusion matrix

Model Evaluation

In [287]:
```
# Print confusion matrix
confusion = metrics.confusion_matrix(y_test, y_pred)
print(confusion)
```

```
[[ 29547   5191]
 [  4003 117154]]
```

In [288]:
```
# check roc score of prediction
auc = metrics.roc_auc_score(y_test, y_pred)
print('AUC: %.3f' % auc)
```

```
AUC: 0.909
```

In [289]:
```
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred)
```

In [290]:
```
# calculate precision-recall curve
precision, recall, thresholds = metrics.precision_recall_curve(y_test, y_pred)
```

In [291]:
```
# Substituting the value of true positive
TP = confusion[1,1]
# Substituting the value of true negatives
TN = confusion[0,0]
# Substituting the value of false positives
FP = confusion[0,1]
# Substituting the value of false negatives
FN = confusion[1,0]
```

Model Evaluation: Sensitivity and Specificity

Sensitivity and Specificity

In [292]:

```
# Calculating the sensitivity
sensi1 =  TP/(TP+FN)
sensi1
```

Out[292]: 0.9669602251623926

In [293]:

```
# Calculating the specificity
speci1 = TN/(TN+FP)
speci1
```

Out[293]: 0.8505671023087109

Model Evaluation: Precision and Recall

Precision and Recall

In [294]:
```
# Precision = TP / TP + FP
confusion[1,1]/(confusion[0,1]+confusion[1,1])
```
Out[294]: 0.9575708038742899

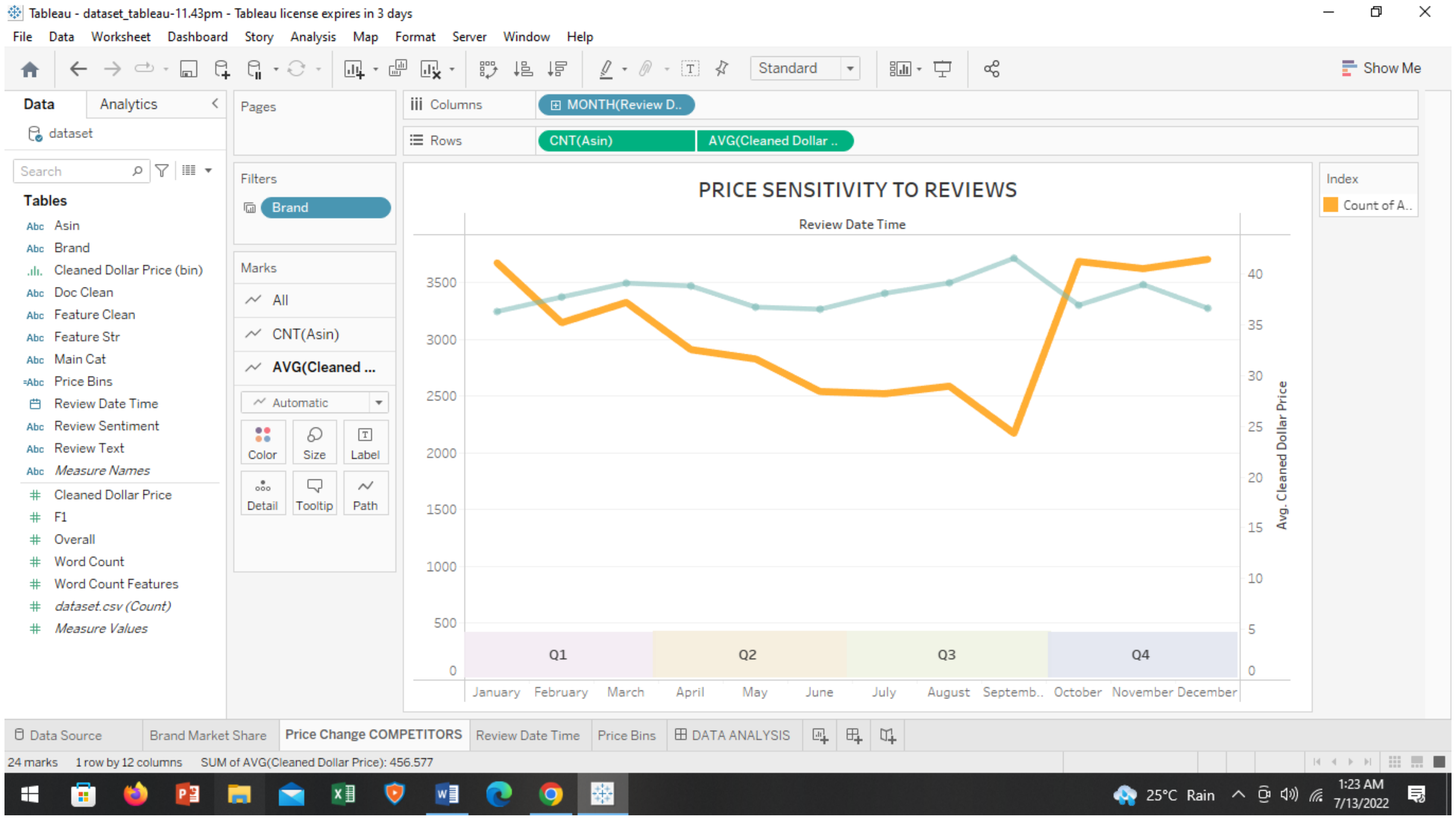
In [295]:
```
#Recall = TP / TP + FN
confusion[1,1]/(confusion[1,0]+confusion[1,1])
```
Out[295]: 0.9669602251623926

Machine Learning Summary

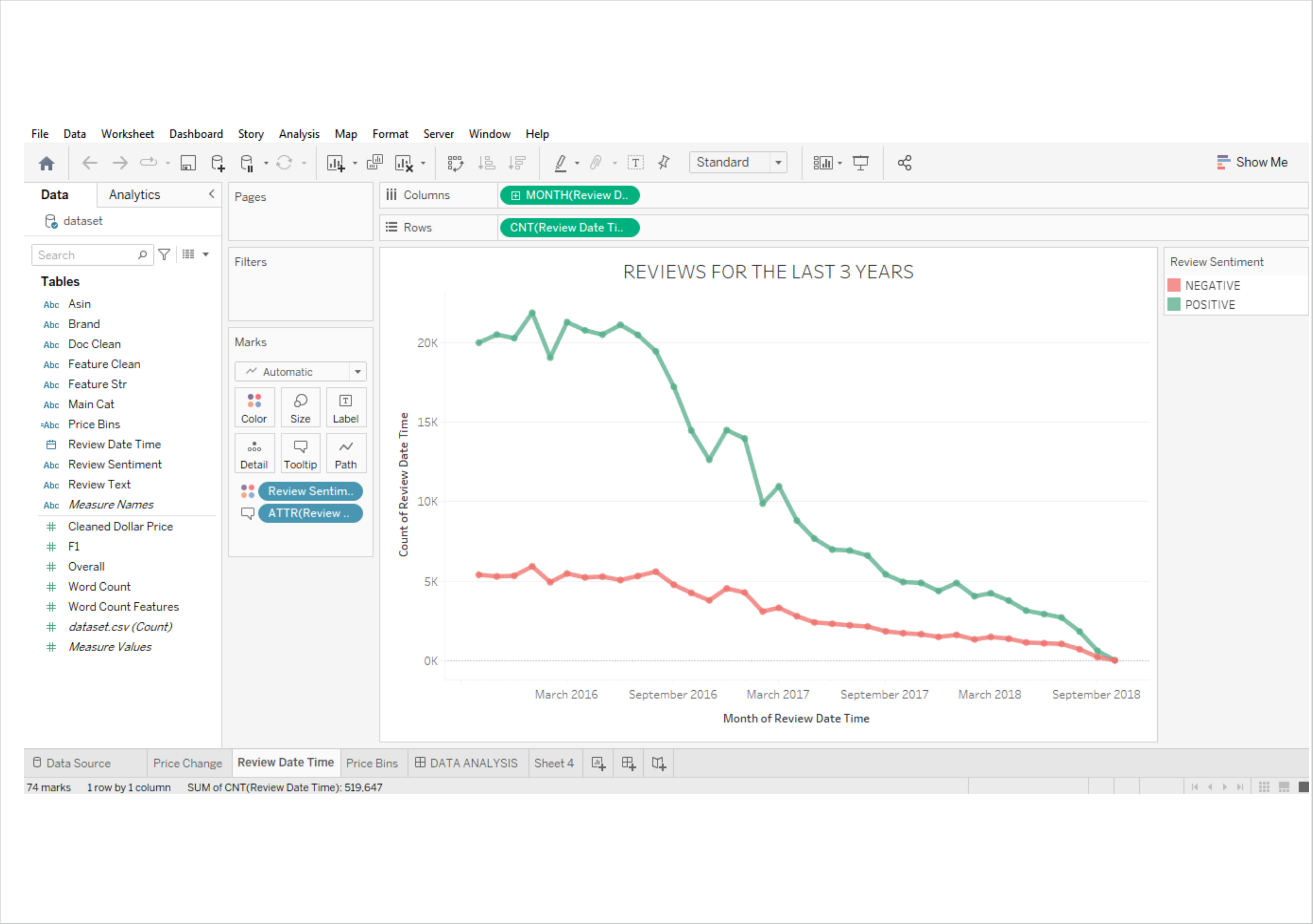
- Trying to find the best algorithm amongst Logistic Regression and Naive Bayes (Multinomial, Complement and Bernoulli)
- Logistic regression and Complement Naïve Bayes algoritms provide best results

Tableau Visualization

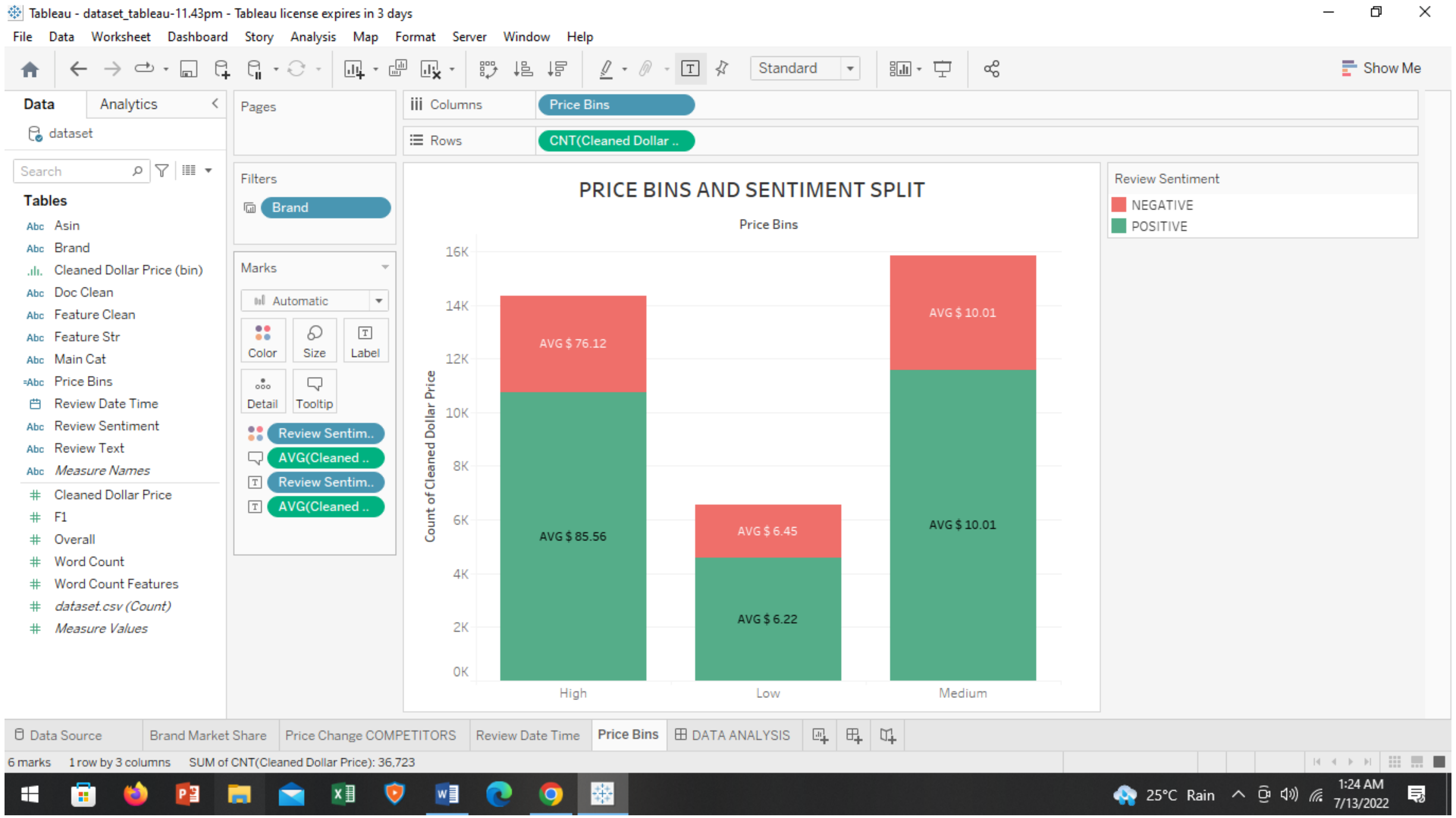
Monthly Distribution of Review and Price



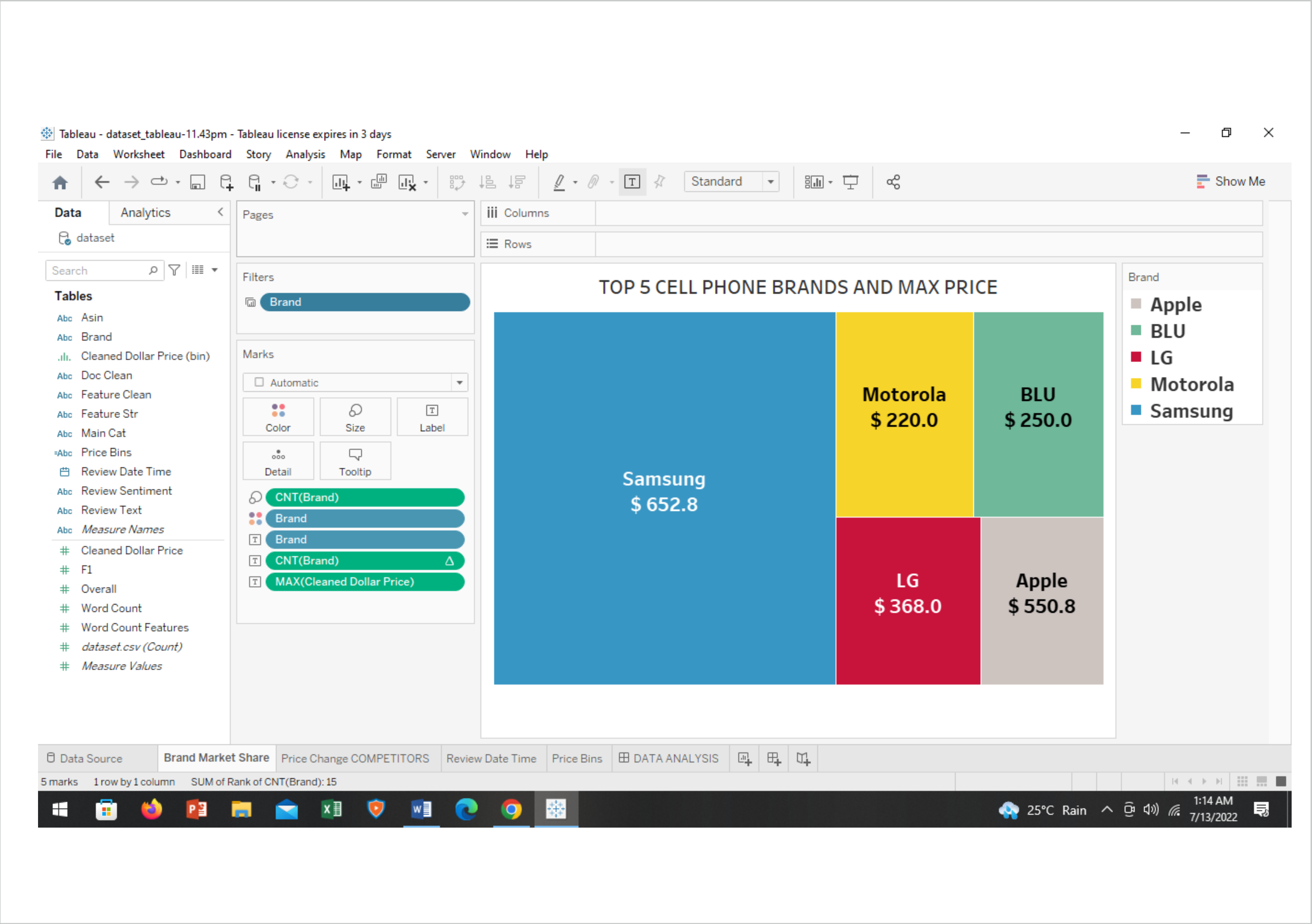
Review Trend for the Past 3 Years



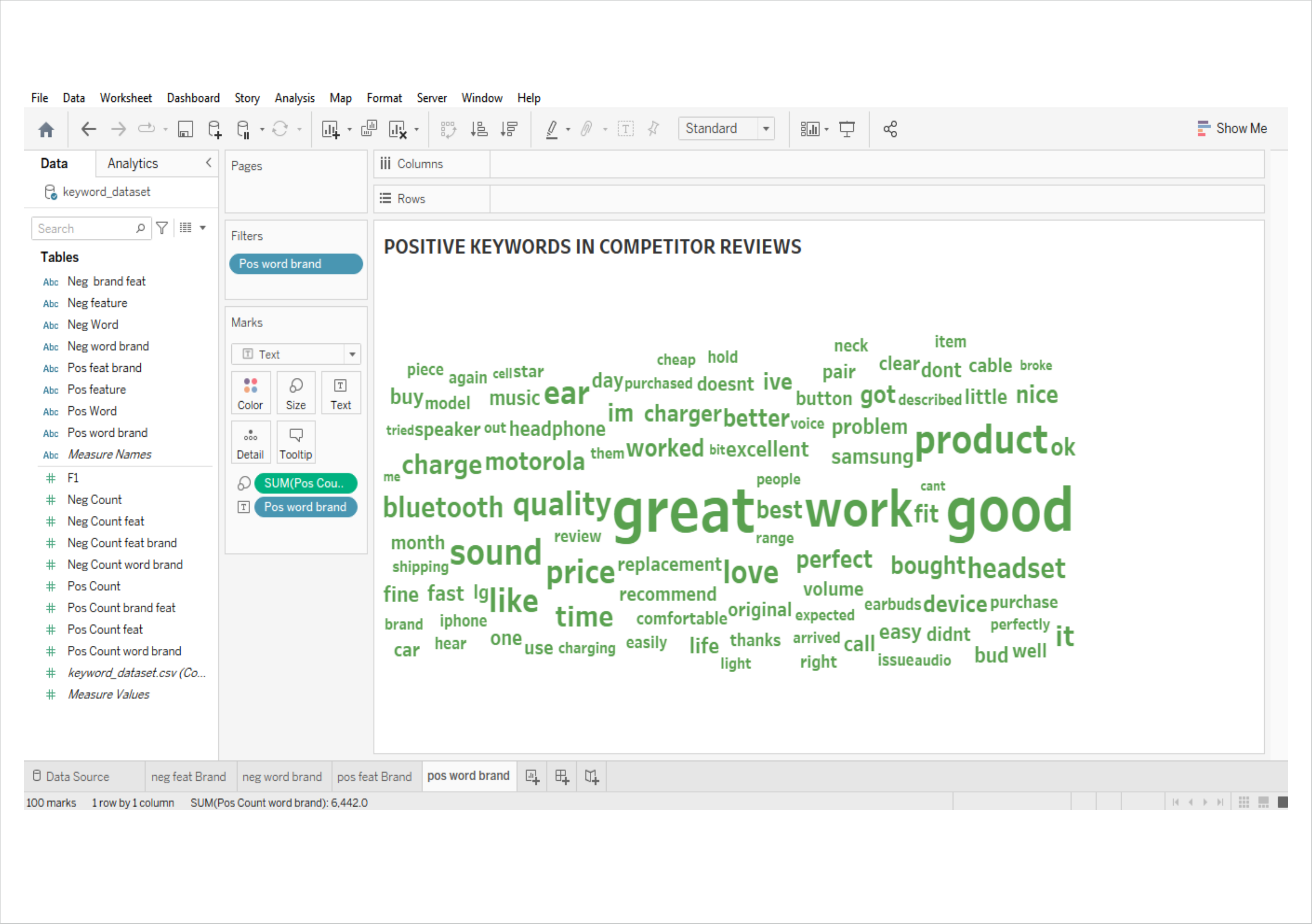
Price Bins Sentiment Split



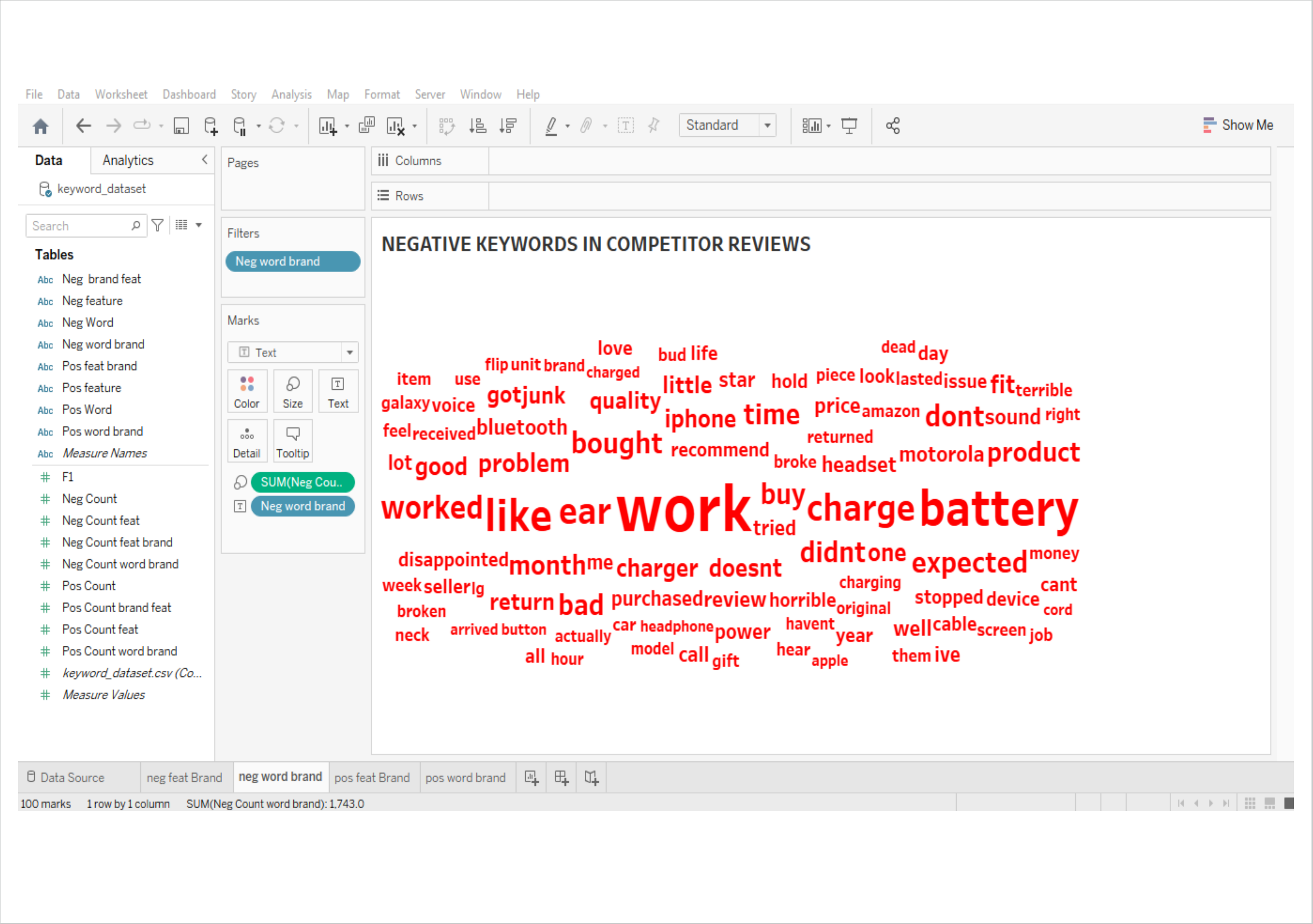
Competitor Brand Market Share and Max Price



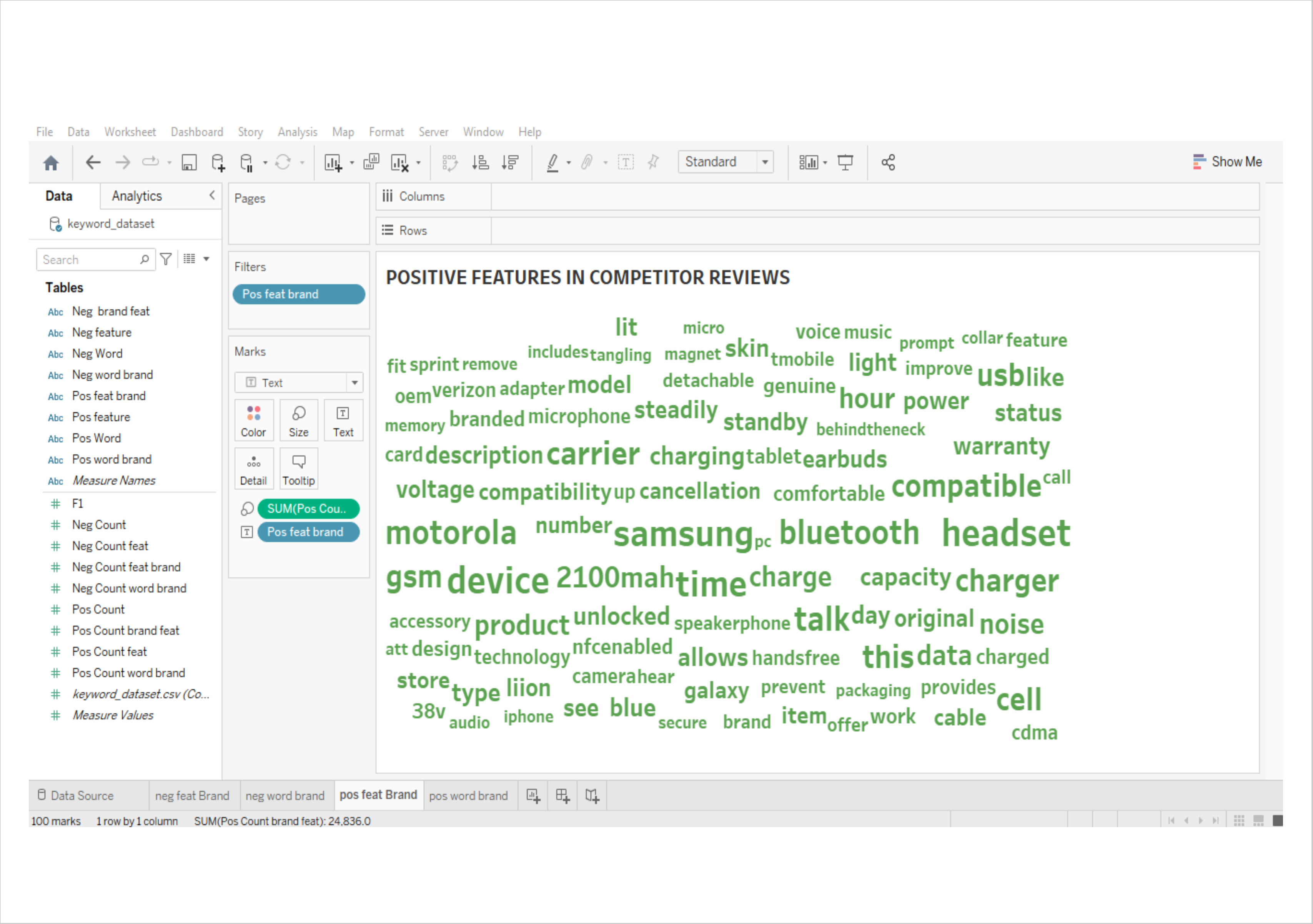
Positive Keywords in only Competitor Reviews



Negative Keywords in only Competitor Reviews



Positive Features of only Competitor Brands



Negative Features of only Competitor Brands