Preprocesamiento y análisis de features

Índice

In [1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
import plotly.express as px

from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
# from gensim.models import Word2Vec
from sklearn.feature_selection import SelectKBest, chi2

import pickle
data = pickle.load(open("saved_feats/data", "rb"))

# FORMATO DE PLOTS
plt.style.use('bmh')
```

Bag of Words

Primero estableceomos la variable sentimiento según el rating, siendo 1-2 negativo, 3 neutro y 4-5 positivo

In [2]:
```
def sentiments(rating):
    if (rating == 5) or (rating == 4):
        return "Positive"
    elif rating == 3:
        return "Neutral"
    elif (rating == 2) or (rating == 1):
        return "Negative"


data_clean = data.dropna(axis='index', how='any')

# Add sentiments to the data
data_text = data_clean["reviews.content"].to_numpy()
data_sentiment = data_clean["reviews.rating"].apply(sentiments).to_numpy()
```

In [3]:
```
data_text
```

Out[3]:
```
array(['Kindle This product so far has not disappointed. My children love to use it and I like the ability to monitor control what content they see with ease.',
       'very fast great for beginner or experienced person. Bought as a gift and she loves it',
       'Beginner tablet for our 9 year old son. Inexpensive tablet for him to use and learn on, step up from t
he NABI. He was thrilled with it, learn how to Skype on it already...',
       ...,
       'Lots of potential!!! I got this to basically experiment with. Straight out of the box, it\'s really no
t that impressive. It\'s a "novelty", however, as I checked out the skills available, lights started coming on
. Literally! Echo\'s ability to integrate with Home Automation is excellent! Having voice commands for turning
 on/off lights may seem like a novelty as well, however, there are endless options for the elderly, or disabled
 with this little assistant, and some customizing...',
       'Great for a "connected home"! This is great for a connected home. People who use this should buy it if
they plan on making everything in their home wi-fi enabled. Otherwise it\'s an expensive SIRI.',
       'Cool toy Cool product. Amazon does a cool job with it. Great audio quality and like the Philips Hue in
tegration.'],
      dtype=object)
```

In [4]:
```
data_sentiment
```

Out[4]:
```
array(['Positive', 'Positive', 'Positive', ..., 'Positive', 'Neutral',
       'Positive'], dtype=object)
```

Vamos a calcular el percentaje de frecuencia que aparece cada sentimiento para ser usado más adelante

In [5]:
```
uniques, counts = np.unique(data_sentiment, return_counts=True)
percentages = dict(zip(uniques, counts * 100 / len(data_sentiment)))
percentages
```

Out[5]:
```
{'Negative': 2.1828605827193886,
 'Neutral': 4.6009113628634132,
 'Positive': 93.21622811446649}
```

Convertimos el contenido en vectores

In [6]:
```
BoW = CountVectorizer(stop_words='english', ngram_range=(1, 1))
data_text_bow = BoW.fit_transform(data_text)

print('Shape of data_text_bow is {}'.format(data_text_bow.get_shape()))
data_text_bow
```

Out[6]:
```
Shape of data_text_bow is (27212, 12067)
<27212x12067 sparse matrix of type '<class 'numpy.int64'>'
	with 375535 stored elements in Compressed Sparse Row format>
```

TFIDF

Suprimimos la redundancia que pueden ocasionar palabras que tienen poco significado y que son las más repetidas en textos largos

In [7]:
```
tfidf = TfidfTransformer(use_idf=True, norm='l2',
                         smooth_idf=True, sublinear_tf=False)
data_text_tfidf = tfidf.fit_transform(data_text_bow)

print('shape of data_text_tfidf is {}'.format(data_text_tfidf.get_shape()))
```

```
shape of data_text_tfidf is (27212, 12067)
```

Selección de mejores features

Seleccionamos las N mejores features según su relación con el outcome

In [8]:
```
best = SelectKBest(chi2, k=5000)
data_text_kbest = best.fit_transform(data_text_tfidf, data_sentiment)

print('shape of data_text_kbest is {}'.format(data_text_kbest.get_shape()))
```

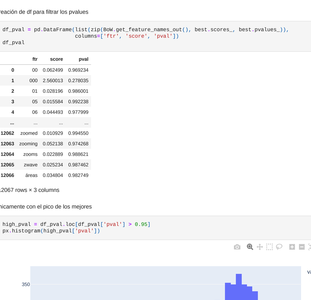
```
shape of data_text_kbest is (27212, 5000)
```

In [9]:
```
print(best.pvalues_.shape)
```

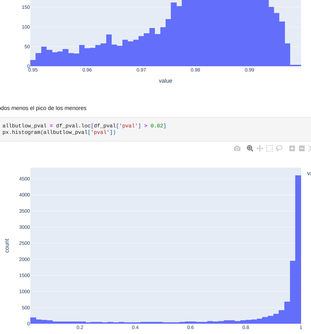
```
(12067,)
```

Queremos ver las features mejor correladas con la variable outcome. Con el BestK hemos seleccionado 5000 de las 12067. Con el histograma de pvalues vemos que salvo las 2000 primeras (p-value < 0.05 => no estan correladas), el resto son buenas y están correladas con el outcome.

In [10]:
```
px.histogram(best.pvalues_[best.scores_ <= 100])
```



In [11]:
```
px.scatter(x=best.pvalues_, y=best.scores_)
```



Voy a hacer zoom en los extremos para poder acotar el valor de pvalue en el que empiezan los picos

In [12]:
```
best_high = [i for i in best.pvalues_ if i > 0.9]
plt.hist(best_high)

best_low = [i for i in best.pvalues_ if i < 0.1]
plt.hist(best_low)
```

Out[12]:
```
(array([822., 191.,  100.,   85.,   77.,   47.,   53.,   53.,   45.,   56.]),
 array([2.93496319e-128, 9.99999638e-003, 1.99819928e-002, 2.99729891e-002,
        3.99639854e-002, 4.99548810e-002, 5.99459783e-002, 6.99369747e-002,
        7.99279711e-002, 8.99189674e-002, 9.99099638e-002]),
 <BarContainer object of 10 artists>)
```

Creación de df para filtrar los pvalues

In [13]:
```
df_pval = pd.DataFrame(list(zip(BoW.get_feature_names_out(), best.scores_, best.pvalues_)),
                       columns=['ftr', 'score', 'pval'])
df_pval
```

Out[13]:

	ftr	score	pval
0	00	0.062499	0.969234
1	000	2.560013	0.278035
2	01	0.028196	0.986001
3	05	0.015584	0.992238
4	06	0.044493	0.977999
...	...	...	...
12062	zoomed	0.010929	0.994550
12063	zooming	0.052138	0.974268
12064	zoom	0.023089	0.988521
12065	zwave	0.025234	0.987462
12066	áreas	0.034804	0.982749

12067 rows × 3 columns

Unicamente con el pico de los mejores

In [14]:
```
high_pval = df_pval.loc[df_pval['pval'] > 0.95]
px.histogram(high_pval['pval'])
```


Todos menos el pico de los menores

In [15]:
```
allbutlow_pval = df_pval.loc[df_pval['pval'] > 0.02]
px.histogram(allbutlow_pval['pval'])
```


In [ ]:
```
df_data_trans = pd.DataFrame(data_text_tfidf.toarray())
# obtenemos la matriz de correlación para ver si hay features que sean redundantes
corr = df_data_trans.corr()
# vemos los límites de los valores de correlación para luego mostrar el heatmap con mas contraste
px.histogram(corr)
```

In [ ]:
```
px.imshow(corr)
```

Word2Vec

In [ ]:
```
preprocessed_reviews = data["reviews.content"].values
train_sentence = [rev.split() for rev in preprocessed_reviews]
# min_count = 5 considers only words that occured atleast 5 times
# size = length of vector
w2v_model_train = Word2Vec(train_sentence, vector_size=100, min_count=5, workers=4)
w2v_words = list(w2v_model_train.wv.index_to_key)
print(w2v_words)
```

In [ ]:
```
w2v_model_train.wv.similarity('zoomed', 'zwave')
```

Almacenamiento de features

In [16]:
```
pickle.dump(data_text_kbest, open("saved_feats/data_text_kbest", "wb"))
pickle.dump(data_sentiment, open("saved_feats/data_sentiment", "wb"))
pickle.dump(data_text, open("saved_feats/data_text", "wb"))
```