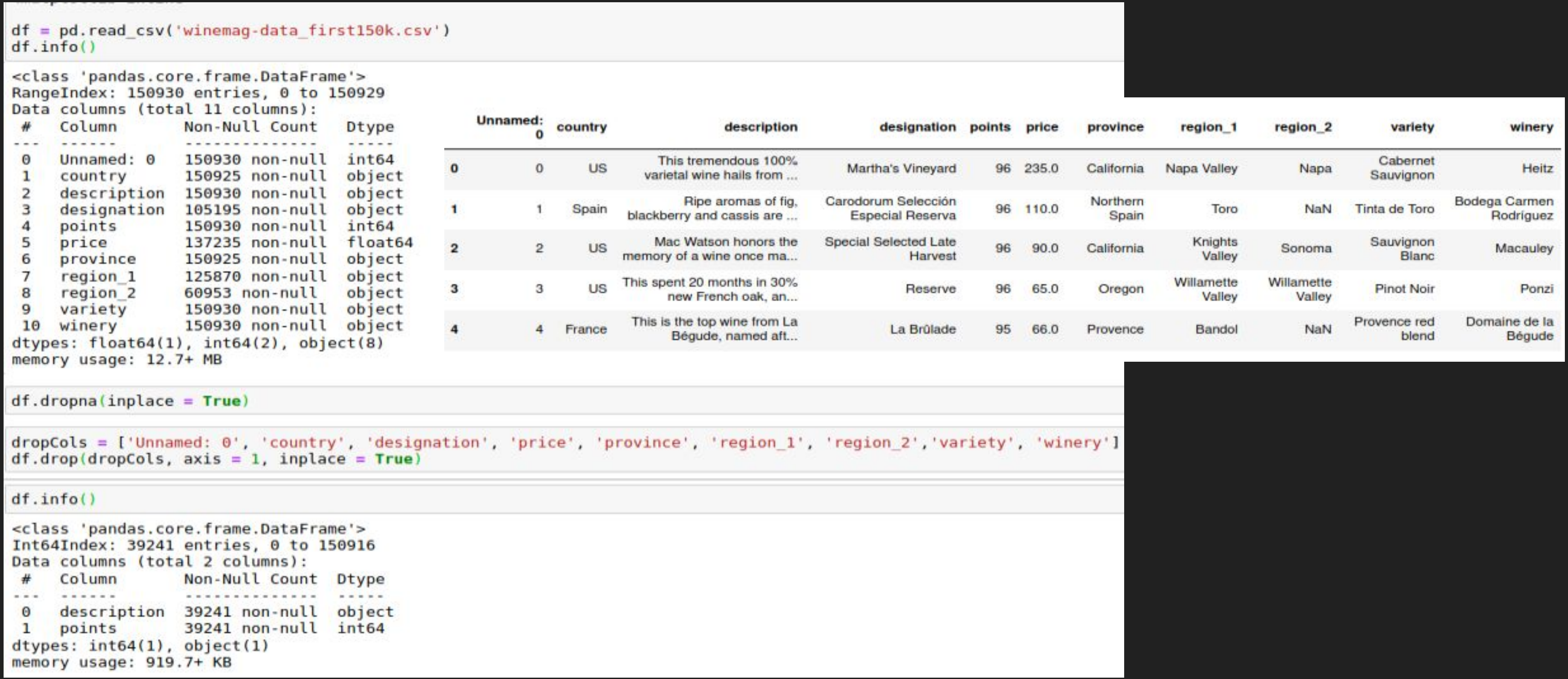
Weka Sentiment Analysis Extension

Alexander Spivey

The Dataset: Wine Review dataset



```
df = pd.read_csv('winemag-data_first150k.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150930 entries, 0 to 150929
Data columns (total 11 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Unnamed: 0   150930 non-null  int64
 1   country      150925 non-null  object
 2   description  150930 non-null  object
 3   designation  105195 non-null  object
 4   points       150930 non-null  int64
 5   price        137235 non-null  float64
 6   province     150925 non-null  object
 7   region_1     125870 non-null  object
 8   region_2     60953 non-null   object
 9   variety      150930 non-null  object
 10  winery       150930 non-null  object
dtypes: float64(1), int64(2), object(8)
memory usage: 12.7+ MB
```

	Unnamed: 0	country	description	designation	points	price	province	region_1	region_2	variety	winery
0	0	US	This tremendous 100% varietal wine hails from ...	Martha's Vineyard	96	235.0	California	Napa Valley	Napa	Cabernet Sauvignon	Heitz
1	1	Spain	Ripe aromas of fig, blackberry and cassis are ...	Carodorum Selección Especial Reserva	96	110.0	Northern Spain	Toro	NaN	Tinta de Toro	Bodega Carmen Rodríguez
2	2	US	Mac Watson honors the memory of a wine once ma...	Special Selected Late Harvest	96	90.0	California	Knights Valley	Sonoma	Sauvignon Blanc	Macauley
3	3	US	This spent 20 months in 30% new French oak, an...	Reserve	96	65.0	Oregon	Willamette Valley	Willamette Valley	Pinot Noir	Ponzi
4	4	France	This is the top wine from La Bégude, named aft...	La Brûlade	95	66.0	Provence	Bandol	NaN	Provence red blend	Domaine de la Bégude

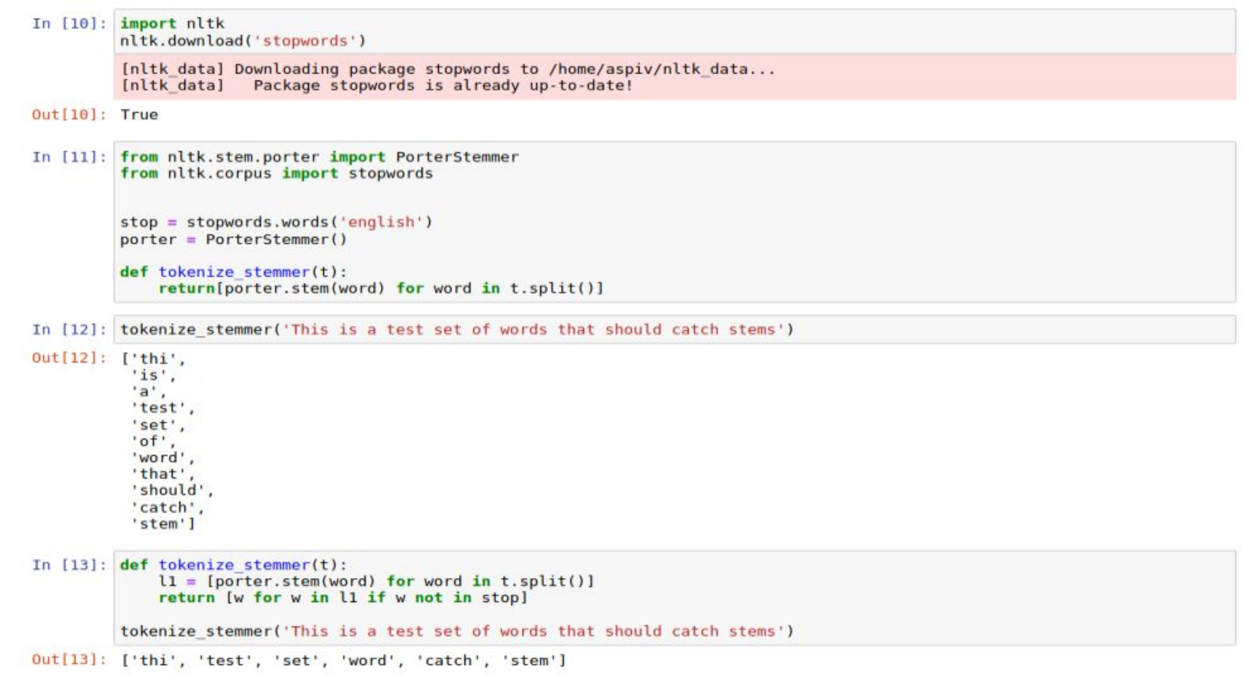
```
df.dropna(inplace = True)
```

```
dropCols = ['Unnamed: 0', 'country', 'designation', 'price', 'province', 'region_1', 'region_2','variety', 'winery']
df.drop(dropCols, axis = 1, inplace = True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 39241 entries, 0 to 150916
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   description  39241 non-null  object
 1   points       39241 non-null  int64
dtypes: int64(1), object(1)
memory usage: 919.7+ KB
```

Tokenizing the Entries



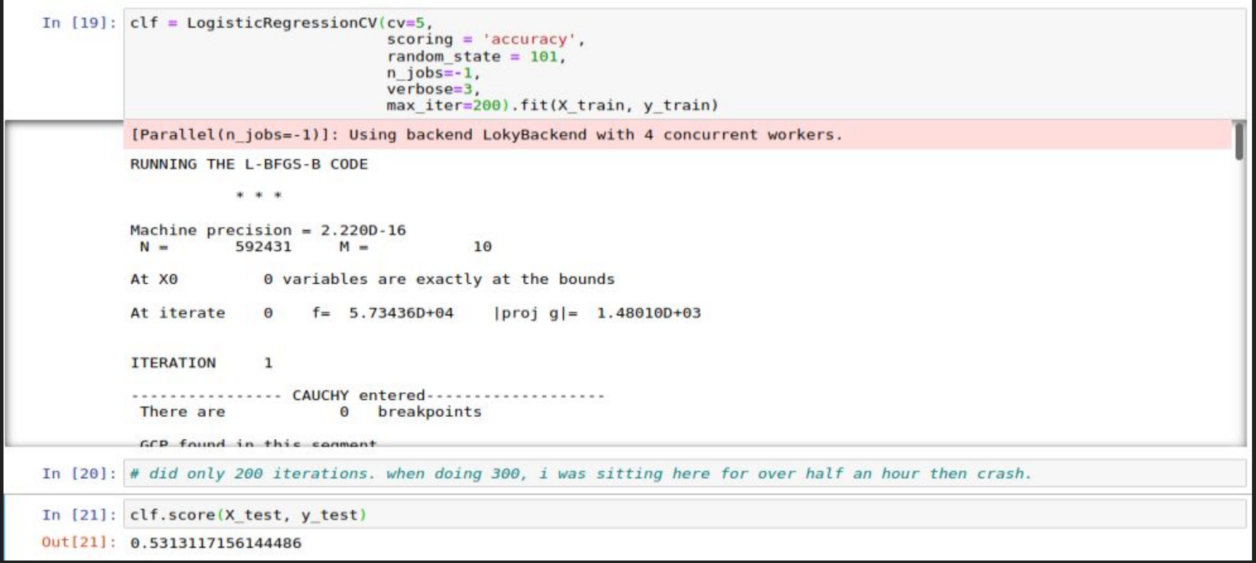
```
In [10]: import nltk
         nltk.download('stopwords')

         [nltk_data] Downloading package stopwords to /home/aspiv/nltk_data...
         [nltk_data]   Package stopwords is already up-to-date!

Out[10]: True
```

```
In [11]: from nltk.stem.porter import PorterStemmer
         from nltk.corpus import stopwords


         stop = stopwords.words('english')
         porter = PorterStemmer()

         def tokenize_stemmer(t):
             return[porter.stem(word) for word in t.split()]
```

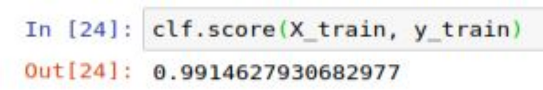
```
In [12]: tokenize_stemmer('This is a test set of words that should catch stems')

Out[12]: ['thi',
          'is',
          'a',
          'test',
          'set',
          'of',
          'word',
          'that',
          'should',
          'catch',
          'stem']
```

```
In [13]: def tokenize_stemmer(t):
             l1 = [porter.stem(word) for word in t.split()]
             return [w for w in l1 if w not in stop]

         tokenize_stemmer('This is a test set of words that should catch stems')

Out[13]: ['thi', 'test', 'set', 'word', 'catch', 'stem']
```

Result

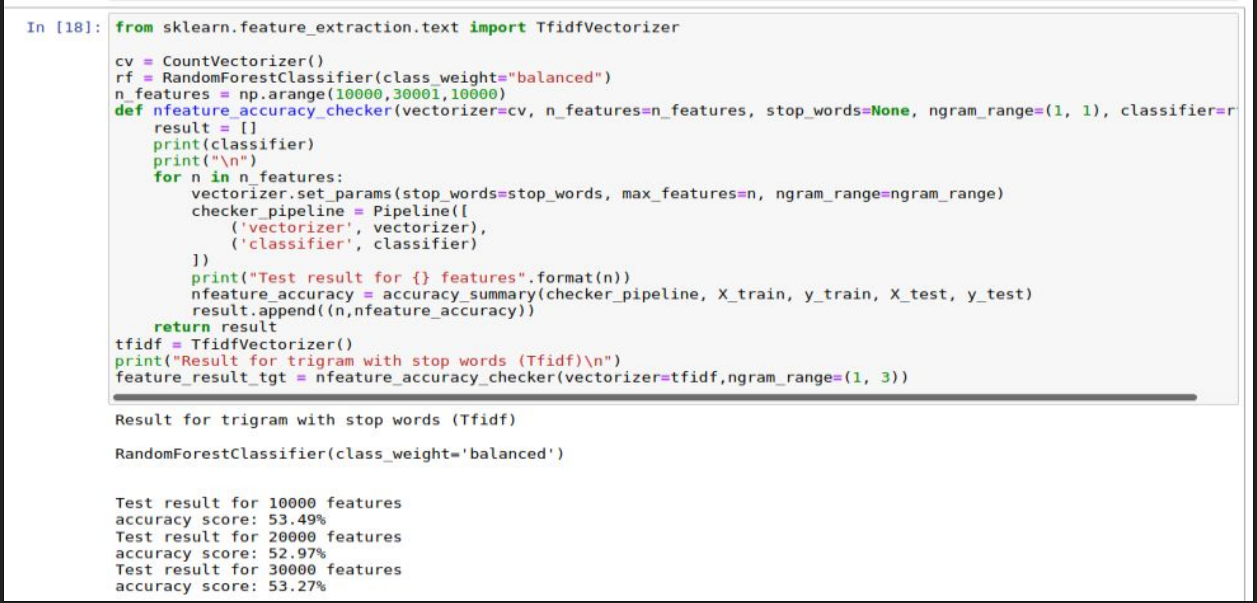


```
In [19]: clf = LogisticRegressionCV(cv=5,
                                    scoring = 'accuracy',
                                    random_state = 101,
                                    n_jobs=-1,
                                    verbose=3,
                                    max_iter=200).fit(X_train, y_train)

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =       592431     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  5.73436D+04    |proj g|=  1.48010D+03


ITERATION     1

---------------- CAUCHY entered-------------------
 There are            0   breakpoints

 GCP found in this segment

In [20]: # did only 200 iterations. when doing 300, i was sitting here for over half an hour then crash.

In [21]: clf.score(X_test, y_test)
Out[21]: 0.5313117156144486
```



```
In [24]: clf.score(X_train, y_train)
Out[24]: 0.9914627930682977
```

Double checking our Result - Random Decision Forest



```
In [18]: from sklearn.feature_extraction.text import TfidfVectorizer

         cv = CountVectorizer()
         rf = RandomForestClassifier(class_weight="balanced")
         n_features = np.arange(10000,30001,10000)
         def nfeature_accuracy_checker(vectorizer=cv, n_features=n_features, stop_words=None, ngram_range=(1, 1), classifier=r
             result = []
             print(classifier)
             print("\n")
             for n in n_features:
                 vectorizer.set_params(stop_words=stop_words, max_features=n, ngram_range=ngram_range)
                 checker_pipeline = Pipeline([
                     ('vectorizer', vectorizer),
                     ('classifier', classifier)
                 ])
                 print("Test result for {} features".format(n))
                 nfeature_accuracy = accuracy_summary(checker_pipeline, X_train, y_train, X_test, y_test)
                 result.append((n,nfeature_accuracy))
             return result
         tfidf = TfidfVectorizer()
         print("Result for trigram with stop words (Tfidf)\n")
         feature_result_tgt = nfeature_accuracy_checker(vectorizer=tfidf,ngram_range=(1, 3))
```

```
Result for trigram with stop words (Tfidf)

RandomForestClassifier(class_weight='balanced')


Test result for 10000 features
accuracy score: 53.49%
Test result for 20000 features
accuracy score: 52.97%
Test result for 30000 features
accuracy score: 53.27%
```