This notebook is used for the project of NYU 6143 Machine Learning course project.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call dr
```

```
import numpy as np
import pandas as pd
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

import nltk
from nltk import word_tokenize
from collections import defaultdict
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
%cd /content/drive/My\ Drive/NYU\_Courses/6143\_ML
```

```
/content/drive/My Drive/NYU_Courses/6143_ML
```

```
df = pd.read_csv('wine_data.csv')
```

To undo cell deletion use ⌘/Ctrl+M Z or the Undo option in the Edit menu

```
nter.most_common(10))}
df = df[df['variety'].map(lambda x: x in top_10_varieties)]
```

```
description_list = df['description'].tolist()
varietal_list = [top_10_varieties[i] for i in df['variety'].tolist()]
varietal_list = np.array(varietal_list)
```

```
count_vect = CountVectorizer()
x_train_counts = count_vect.fit_transform(description_list)
tfidf_transformer = TfidfTransformer()
x_train_tfidf = tfidf_transformer.fit_transform(x_train_counts)
train_x, test_x, train_y, test_y = train_test_split(x_train_tfidf, varietal_list, t
```

Here it is better to use random grid search to find optimal parameters. However, to simplify the problem, we use the default parameters to show the demo.

```
clf = LogisticRegression(max_iter=1e5, tol=1e-5)
```

```
def get_model_report(clf):
    clf.fit(train_x, train_y)
    y_score = clf.predict(test_x)
    print(classification_report(test_y, y_score))
```

```
get_model_report(LogisticRegression(max_iter=1e5, tol=1e-5))
```

```
                precision    recall  f1-score   support

             0       0.77      0.88      0.83      3945
             1       0.84      0.94      0.89      3510
             2       0.66      0.80      0.72      2761
             3       0.84      0.78      0.81      2707
             4       0.81      0.80      0.81      2083
             5       0.92      0.86      0.89      1582
             6       0.86      0.75      0.80      1496
             7       0.81      0.60      0.69      1271
             8       0.87      0.79      0.83      1045
             9       0.84      0.35      0.49       997

      accuracy                           0.80     21397
     macro avg       0.82      0.75      0.78     21397
  weighted avg       0.81      0.80      0.80     21397
```

```
get_model_report(MultinomialNB(alpha=0.05))
```

```
                precision    recall  f1-score   support
```

To undo cell deletion use ⌘/Ctrl+M Z or the Undo option in the Edit menu

```
             3       0.72      0.75      0.73      2707
             4       0.70      0.71      0.71      2083
             5       0.89      0.76      0.82      1582
             6       0.84      0.62      0.72      1496
             7       0.74      0.35      0.47      1271
             8       0.84      0.71      0.77      1045
             9       0.64      0.08      0.14       997

      accuracy                           0.72     21397
     macro avg       0.74      0.65      0.66     21397
  weighted avg       0.73      0.72      0.70     21397
```

```
1 get_model_report(SVC(kernel='linear'))
```

	precision	recall	f1-score	support
0	0.78	0.88	0.83	3945
1	0.84	0.94	0.89	3510
2	0.65	0.81	0.72	2761
3	0.85	0.77	0.81	2707
4	0.82	0.82	0.82	2083
5	0.92	0.86	0.89	1582
6	0.86	0.75	0.80	1496
7	0.81	0.60	0.69	1271
8	0.87	0.81	0.84	1045
9	0.85	0.34	0.48	997
accuracy			0.80	21397
macro avg	0.83	0.76	0.78	21397
weighted avg	0.81	0.80	0.80	21397

```
1 get_model_report(SVC(kernel='rbf'))
```

	precision	recall	f1-score	support
0	0.78	0.90	0.84	3945
1	0.84	0.95	0.89	3510
2	0.66	0.83	0.73	2761
3	0.87	0.77	0.82	2707
4	0.83	0.83	0.83	2083
5	0.94	0.87	0.90	1582
6	0.91	0.74	0.82	1496
7	0.86	0.59	0.70	1271
8	0.89	0.81	0.85	1045
9	0.93	0.34	0.50	997
accuracy			0.82	21397
macro avg	0.85	0.76	0.79	21397
weighted avg	0.83	0.82	0.81	21397

To undo cell deletion use ⌘/Ctrl+M Z or the Undo option in the Edit menu

utility functions

```
1 def count_top_x_words(corpus, top_x, skip_top_n):
2     count = defaultdict(lambda: 0)
3     for c in corpus:
4         for w in word_tokenize(c):
5             count[w] += 1
6     count_tuples = sorted([(w, c) for w, c in count.items()], key=lambda x: x[1], r
7     return [i[0] for i in count_tuples[skip_top_n: skip_top_n + top_x]]
```

```
    return [i[0] for i in counts_tuples[skip_top_n:skip_top_n+top_n]]


def replace_top_x_words_with_vectors(corpus, top_x):
    topx_dict = {top_x[i]: i for i in range(len(top_x))}
    return [
        [topx_dict[w] for w in word_tokenize(s) if w in topx_dict]
        for s in corpus
    ], topx_dict


def filter_to_top_x(corpus, n_top, skip_n_top=0):
    top_x = count_top_x_words(corpus, n_top, skip_n_top)
    return replace_top_x_words_with_vectors(corpus, top_x)
```

```
from keras.models import Sequential
from keras.layers import Dense, Conv1D, Flatten
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
from keras.utils import to_categorical
```

```
mapped_list, word_list = filter_to_top_x(description_list, 2500, 10)
varietal_list_o = [top_10_varieties[i] for i in df['variety'].tolist()]
varietal_list = to_categorical(varietal_list_o)
```

```
max_review_length = 150
mapped_list = sequence.pad_sequences(mapped_list, maxlen=max_review_length)
train_x, test_x, train_y, test_y = train_test_split(mapped_list, varietal_list, tes
```

```
embedding_vector_length = 64
model = Sequential()

model.add(Embedding(2500, embedding_vector_length, input_length=max_review_length))
model.add(Conv1D(50, 5))
model.add(Flatten())
```



```
tmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy
model.fit(train_x, train_y, epochs=3, batch_size=64)
```

```
Epoch 1/3
781/781 [==============================] - 36s 46ms/step - loss: 1.0562 - accura
Epoch 2/3
781/781 [==============================] - 36s 45ms/step - loss: 0.6288 - accura
Epoch 3/3
781/781 [==============================] - 36s 46ms/step - loss: 0.5017 - accura
<tensorflow.python.keras.callbacks.History at 0x7f81d93ad8d0>
```

```
y_score = model.predict(test_x)
y_score = [[1 if i == max(sc) else 0 for i in sc] for sc in y_score]
```

```
print(classification_report(test_y, y_score))
```

```
              precision    recall  f1-score   support

           0       0.84      0.78      0.81      3995
           1       0.87      0.86      0.86      3527
           2       0.62      0.79      0.69      2796
           3       0.84      0.69      0.76      2657
           4       0.74      0.82      0.78      2104
           5       0.85      0.88      0.86      1584
           6       0.76      0.75      0.76      1496
           7       0.67      0.60      0.63      1268
           8       0.83      0.76      0.79      1033
           9       0.50      0.46      0.48       937

   micro avg       0.77      0.77      0.77     21397
   macro avg       0.75      0.74      0.74     21397
weighted avg       0.77      0.77      0.77     21397
 samples avg       0.77      0.77      0.77     21397
```