PROJECT REPORT ON

Analysis on Wine Dataset for wine classification

Prepared By: Manisha Khatri

A#:25286693

Under the guidance of:

Dr. Vineetha Menon

CS588:Intro to Big Data Computing

Fall 2019

Computer Science Department

University of Alabama Huntsville

Table of Contents

Introduction

Imagine there is a special occasion or just an unplanned evening date, and you decide to celebrate it with a bottle of wine. Imagine having to choose from more than 10000 varieties of wines? Whether you're an expert or a newbie, it can be a task to identify good wines with tons of information available online. One would have to read through countless reviews, that could still leave them confused. To make our lives easy, big data techniques can be used to process tons of information available to identify whether a wine is worth it or not. The scope of the project includes to classify the wines from different countries as Best, Better, good or bad based on the reviews, ratings, price and country. The data for the project was scraped from WineEnthusiast during the week of June 15th, 2017.[1] The project can be further extended to develop a wine recommender system, that could theoretically identify the wine based on description of requirement.

Dataset Description

The dataset contains more than 130k records and 14 columns. The features of the data with their description are provided in the figure below.

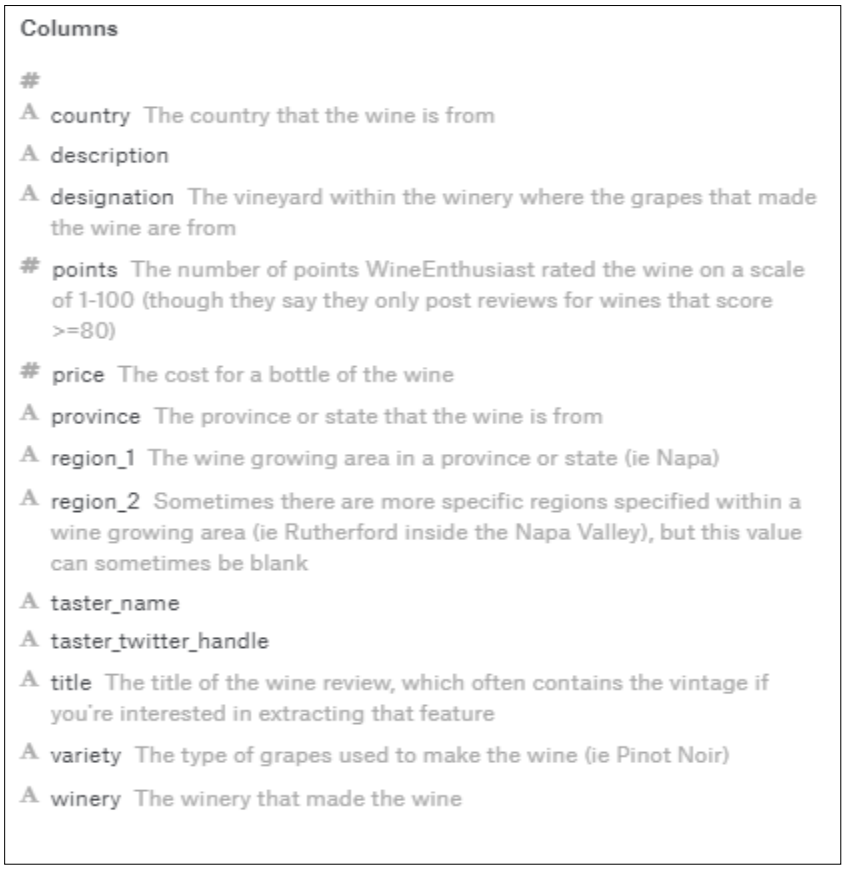

Fig.1: Dataset schema description

From general understanding, from the features shown above following features could contribute while classifying a wine:

1. Description
2. Points
3. country
4. variety
5. title

Data Pre-processing

To ensure any inconsistencies are avoided, the data is preprocessed to remove records containing duplicate, missing and invalid values or characters.

Also there are around 45 different countries and 702 distinct wine varieties in the dataset. To limit the scope of the project, the countries with more than 3764 records and varieties with more than 5000 are considered. This results in subset containing around 40,688 records with 5 countries and 15 wine varieties. The relation between different features is visualized further in the data visualization section.

The data contains textual data in form of description that needs to be specially processed in order to feed it to our predictive model. The text must be parsed to remove words, called tokenization. Then the words need to be encoded as integers or floating point values for use as input to a machine learning algorithm, called feature extraction (or vectorization)[2] .

In the project, Count Vectorizer is used. The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

Data Visualization

Variety Vs Points (47, variety records>300)

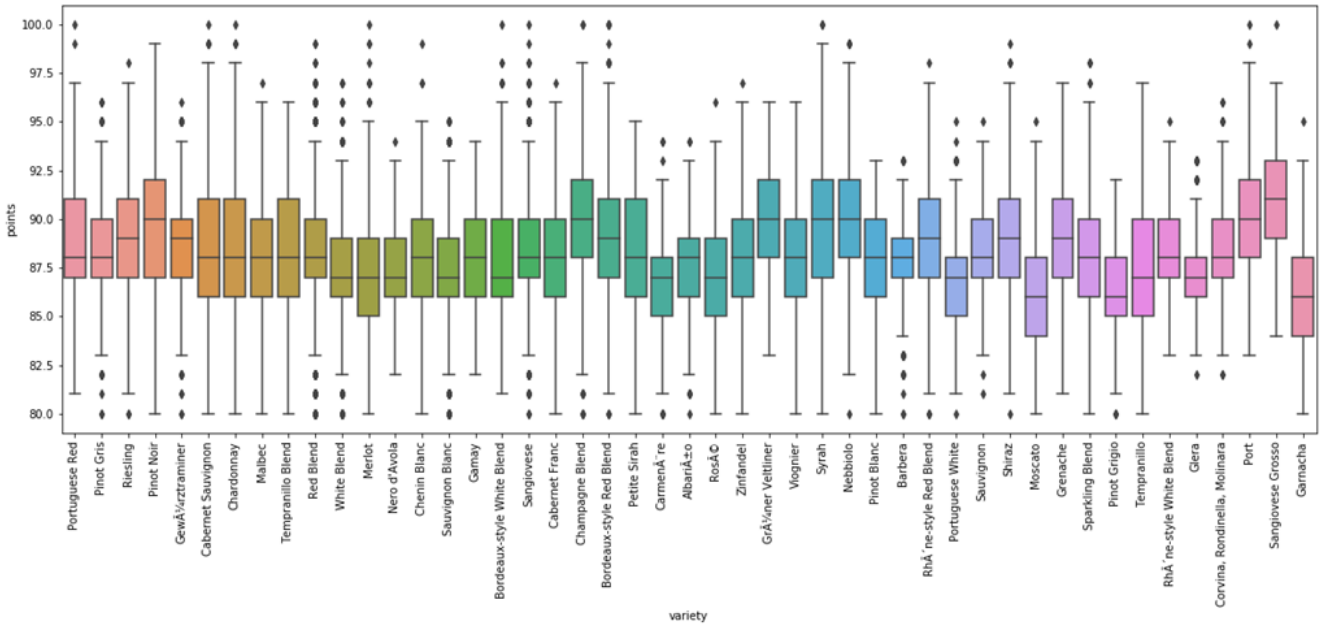

Fig.2:Variety Vs Points (47, variety records>300)

The boxplot above is of wine varieties (containing at least greater than 300 records) and their point distribution. We see that Sangiovese Grosso has the highest (points) median value and Grenache has the lowest median value. Although there are slight variations and dips, the range and distribution are mostly uniform with average median rating around 88.

Country Vs Points

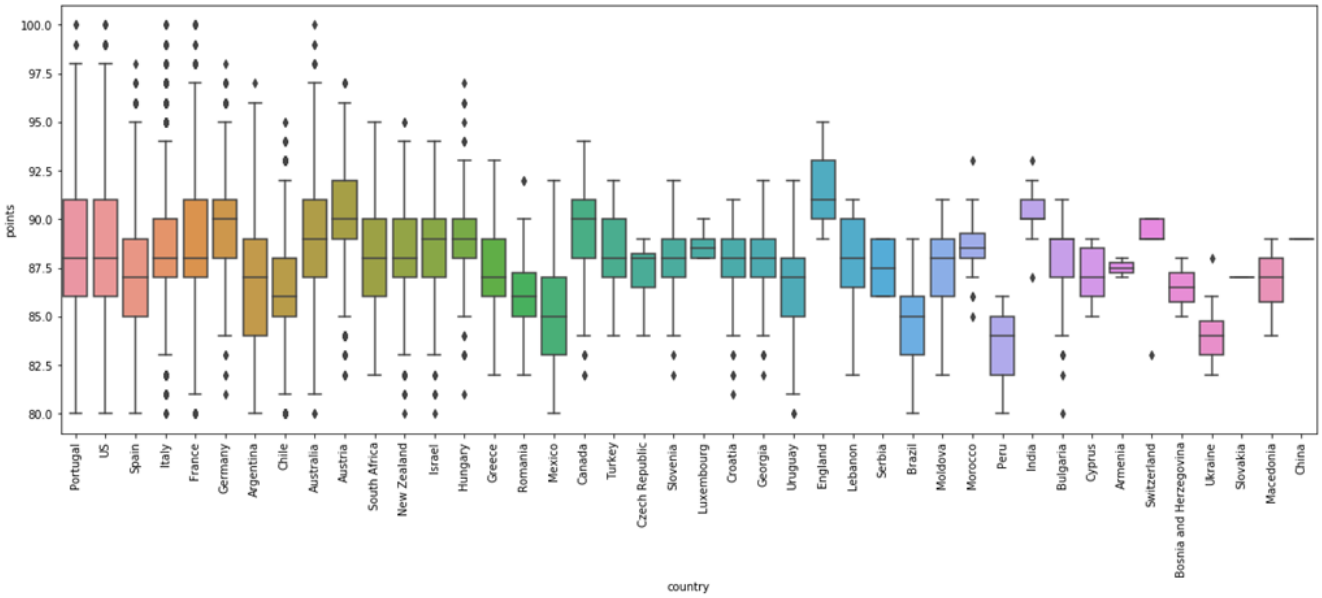

Fig. 3:Country Vs Points

The box plot above indicates that there is variation in range of points for different countries. This can be accounted for the fact that there are fewer samples for some countries. Although the median is around 84-91 points. The countries with few records may not help in building the classification models and therefore are not considered further in the project.

Variety Vs country

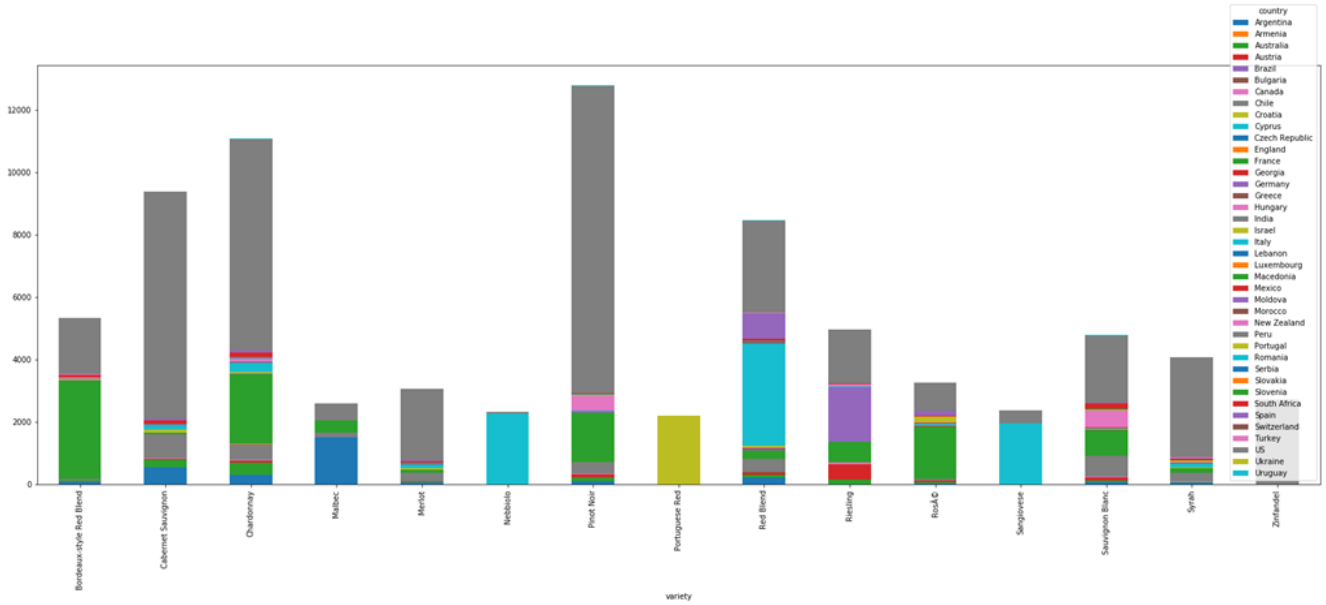

Fig. 4: Variety Vs country

It can be observed that US has maximum number of varieties followed by France and Italy. Also the numbers of records are highest for USA.

Variety Vs Points: (Variety> 2170)

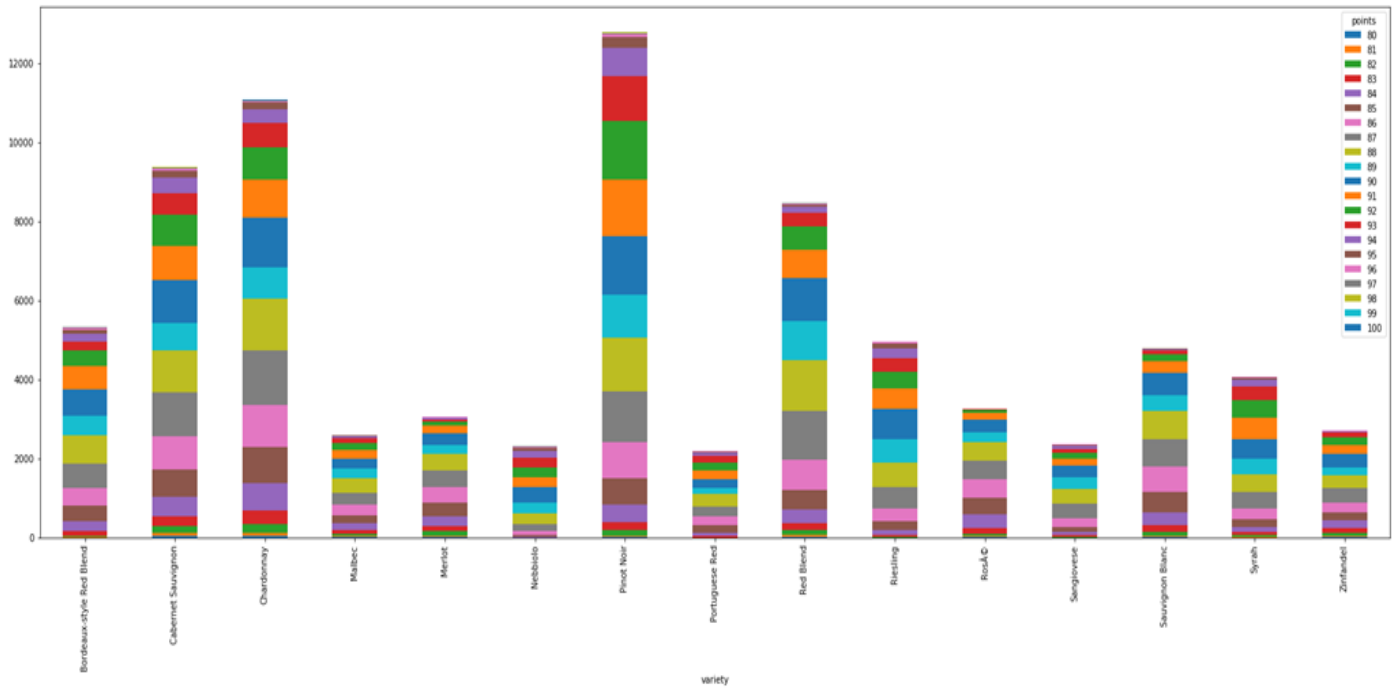

Fig. 5: Variety Vs Points

The bar chart above shows that there are ratings ranging from 80-100 in almost all the wine varieties

Variety vs Country:

dataset: country>4872

variety>2170

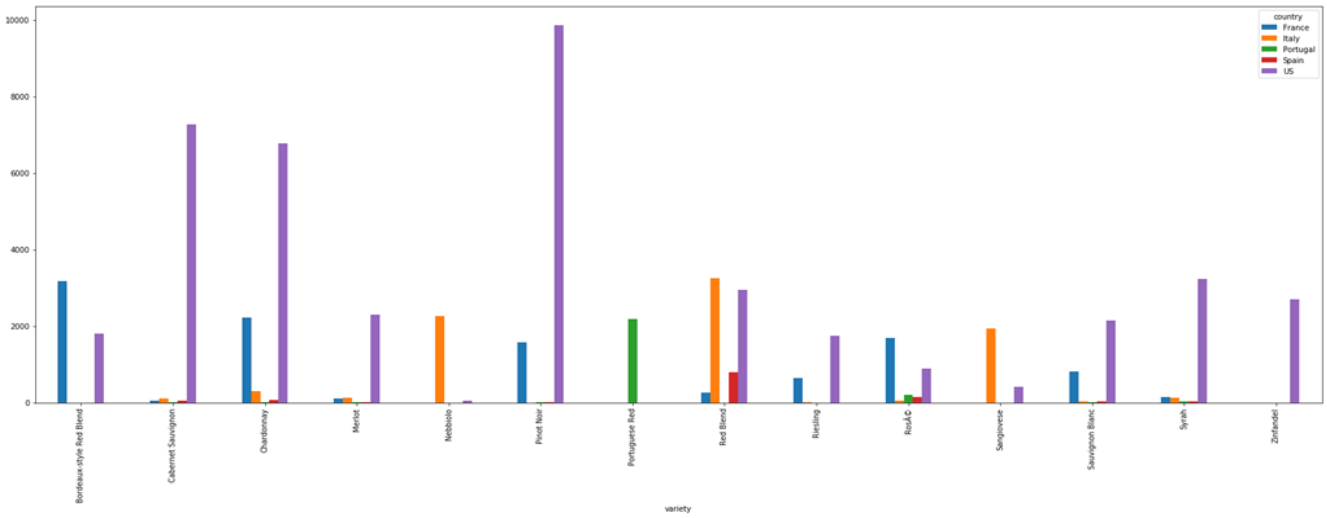

Fig.6: Variety vs Country

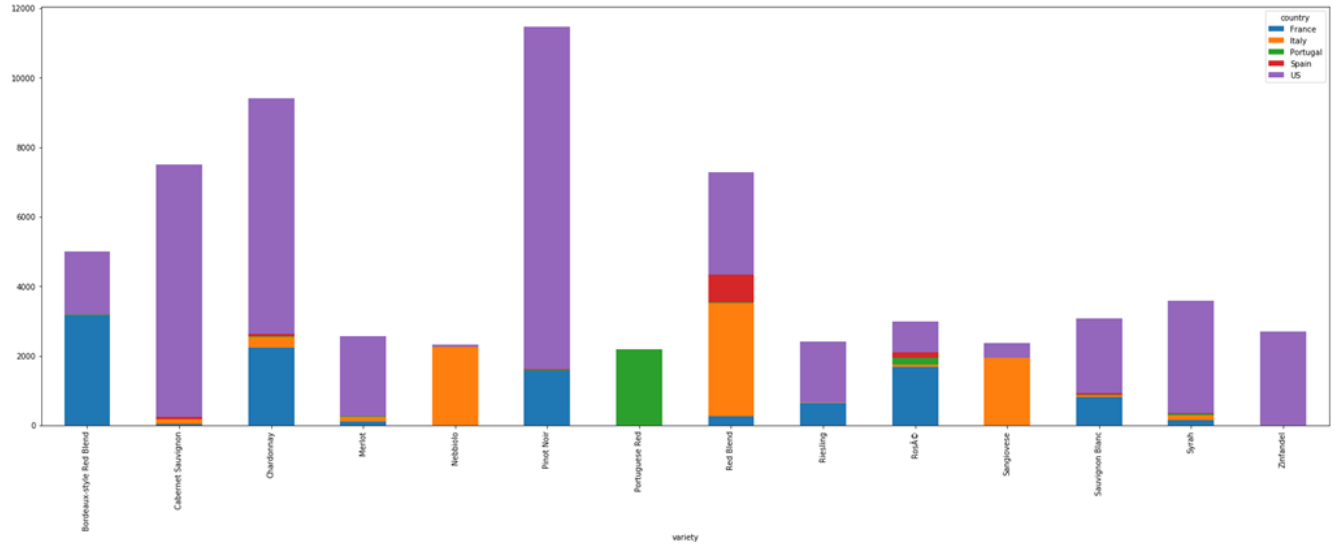

Fig. 7: Variety vs Country

Country Vs Variety

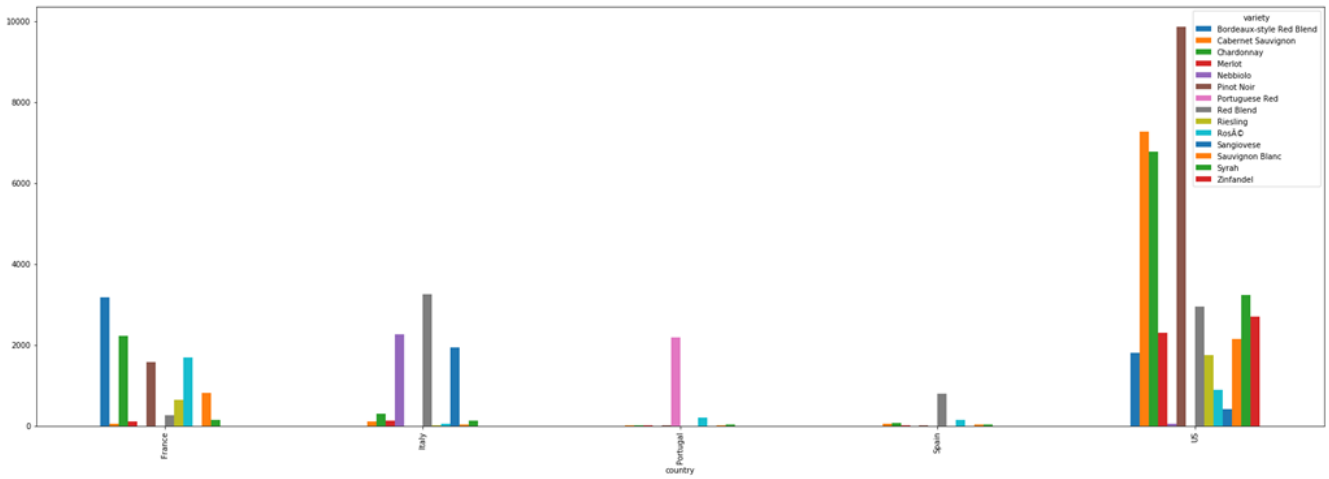

Fig.8: Country Vs Variety

It can be observed that maximum number of wine varieties and their samples is available for USA. Although other countries have relatively samples, they are retained as they have good number of records that can be considered.

Discussion of methods

Vectorization of dataset:

- CountVectorizer: The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.[2] It counts the word frequencies in a document
- TFIDFVectorizer: In TFIDFVectorizer the value increases proportionally to count, but is offset by the frequency of the word in the corpus

To evaluate the performance the accuracy of both the vectorizers for 10%, 30%,50%,70%,90% test data for KNN algorithm(with 3000 n_neighbours) are recorded:

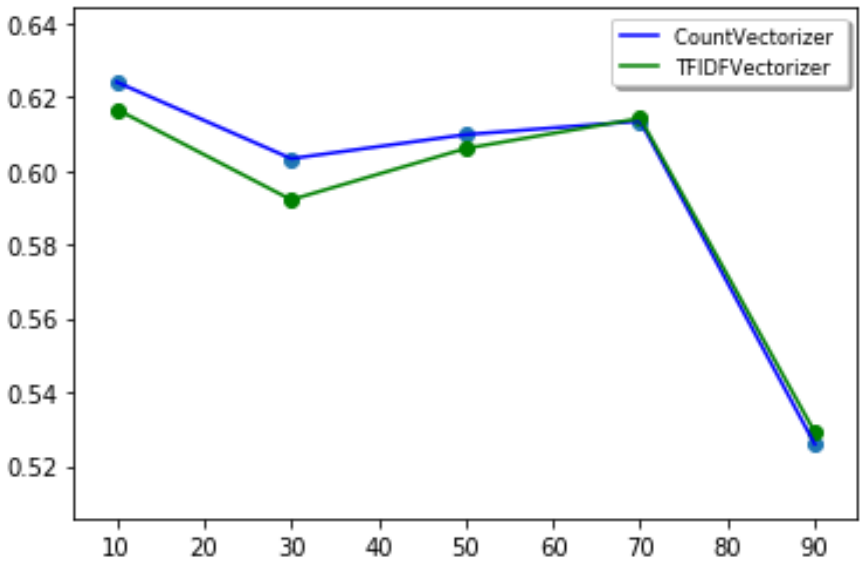

Fig 10. Countvectorizer Vs TFIDFVectorizer

The accuracy is almost similar but slightly more for count vectorizer in case of wine dataset. Therefore, Countvectorizer is used.

Further, class labels are generated from points/ratings feature (ranges from 80-100) as follows:

- Points<=85 : "Bad"
- Points<=90: "Good"
- Points<=95: "Better"
- Points<=100: "Best"

A classification model is built on the vectorized description, country, points and price based on Multinomial Naive Bayes(MNB), K- nearest Neighbor(k-NN) and Support Vector Machines(SVM)

Dimensionality reduction:

Techniques such as PCA and LDA are used to observe the performance of the model.

PCA: It is an unsupervised linear transformation technique for feature extraction and dimensionality reduction. PCA helps us to identify patterns in data based on the correlation between features. In a nutshell, PCA aims to find the directions of maximum variance in high-dimensional data and projects it onto a new subspace with equal or fewer dimensions than the original one.[3]

LDA: It finds a new feature space to project the data in order to maximize class separability and minimizes inter-class variability.

Classification Algorithms:

To classify the wines following 3 classification algorithms are used:

1. K-Nearest Neighbor
2. Support Vector machines
3. Naive Bayes

K-Nearest Neighbor: KNN (K-Nearest Neighbor) is a simple supervised classification algorithm that implements the k-nearest neighbors vote. It computes the distance(Manhattan , Euclidean etc) between the new data point with every training example. Model picks K entries in the database which are closest to the new data point. Then it does the majority vote i.e the most common class/label among those K entries will be the class of the new data point.

Support Vector machines: The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence [3]

Naive Bayes classification: A Gaussian Naive Bayes algorithm is a type of NB algorithm used when the features have continuous values. It's also assumed that all the features are following a gaussian distribution i.e, normal distribution. It predicts the probability of each class based on the feature vector for text classification for continuous big data with a prior distribution of the probability, tackling the challenges of the curse of the dimensionality [4]

Results and analysis

The variance captured by the first principal component after PCA as given by the explained variance is 99.1%. This implies that maximum data is captured by the first PC

The above mentioned classification techniques are applied to preprocessed dataset, with 10%, 30%, 50%, 70% and 90% test data. The performance is measured by the accuracy attribute and plotted on the graph below.

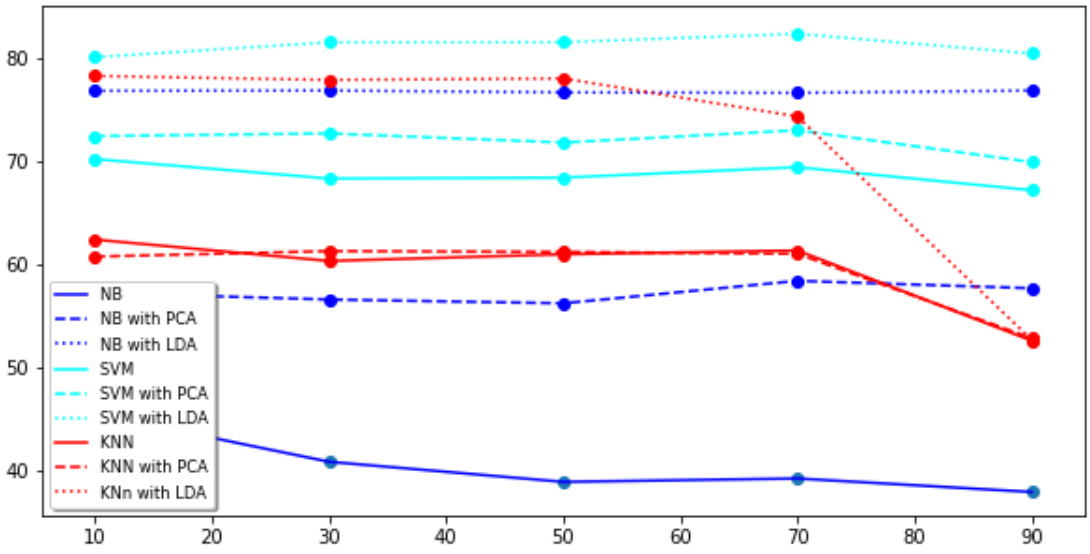

Fig. 11: Comparison graph of classifier

The accuracy is around 75%-80% after LDA and around 60 % after PCA. The performance of KNN after PCA did not improve. Without dimensionality reduction the naive bayes classifier gave very low accuracy of around 40% -45%. The accuracy of SVM is around 69% without dimensionality reduction

Conclusion

It is observed that all the 3 classifiers perform better after dimensionality reduction, specifically LDA. The performances are comparable between Naive bayes and KNN, but KNN's performance is slightly better. SVM gives the highest accuracy of classification of wine into the respective class labels.

References

[1] Wine Reviews

https://www.kaggle.com/zynicide/wine-reviews

[2]Machine learning Mastery

https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/

[3] Towards Data Science

https://towardsdatascience.com/

[4]Medium

https://medium.com/@gp_pulipaka/applying-gaussian-na%C3%AFve-bayes-classifier-in-python-part-one-9f82aa8d9ec4

Appendix:

Source code:

```
import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

from sklearn.feature_extraction import text

from sklearn.decomposition import PCA

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

from sklearn.feature_extraction.text import CountVectorizer

# splitting the data set into training set and test set

from sklearn.model_selection import train_test_split

#Import knearest neighbors Classifier model

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion_matrix

from sklearn.metrics import accuracy_score

from nltk.tokenize import RegexpTokenizer

from nltk.stem.snowball import SnowballStemmer

from sklearn.naive_bayes import GaussianNB

from sklearn.svm import SVC

#from sklearn.model_selection import GridSearchCV


#load dataset
```

```
data = pd.read_csv('F:\study/BDA/project/bdaproject/BDA project/Datasets/Edited-winemag-data-130k-v2.csv',sep=',')


#handle varieties with special characters

data = data.loc[data['variety'].str.contains(r'[^\x00-\x7F]+') == False]


#creating subset with sufficient samples for countries and variety

data = data.groupby('country').filter(lambda x: len(x) >3764)

data = data.groupby('variety').filter(lambda x: len(x) >5000).reset_index()


#vectorize country

df_country = data.country

country = ['US','Italy','France','Spain','Portugal']

k=0

for i in country:

    df_country=df_country.replace(i, k)

    k=k+1


#vectorize variety

df_variety=data.variety

variety_names=["Pinot Noir","Chardonnay","Cabernet Sauvignon","Red Blend","Bordeaux-style Red Blend","Riesling","Sauvignon
Blanc","Syrah","RosÃ©","Merlot","Zinfandel","Malbec","Sangiovese","Nebbiolo","Portuguese Red"]

k=0

for i in variety_names:

    df_variety=df_variety.replace(i, k)

    k=k+1
```

```
#vectorize points
df_review = data.points
for i in range(80,101):
    if(i<=85):
        df_review=df_review.replace(i,'Bad')
    elif(i<=90):
        df_review=df_review.replace(i,'Good')
    elif(i<=95):
        df_review=df_review.replace(i,'Better')
    else:
        df_review=df_review.replace(i,'Best')

#kernel
punc = ['.', ',', '"', "'", '?', '!', ':', ';', '(', ')', '[', ']', '{', '}',"%"]
stop_words = text.ENGLISH_STOP_WORDS.union(punc)
desc = data['description'].values
print("data variety:",data['variety'].count())
print("description count:",data['description'].count())


stemmer = SnowballStemmer('english')
tokenizer = RegexpTokenizer(r'[a-zA-Z\']+')
```

```
def tokenize(text):
    return [stemmer.stem(word) for word in tokenizer.tokenize(text.lower())]

# creating bag of words model
cv = CountVectorizer(stop_words = stop_words, tokenizer = tokenize, max_features = 3000)
X = cv.fit_transform(desc).toarray()
#cv._validate_vocabulary()
#print('loaded_vectorizer.get_feature_names(): {0}'.format(cv.get_feature_names()))
X = pd.DataFrame(X)
print(X)

df_gth = data.filter(['price'], axis=1)
df_gth['Variety']=df_variety
df_gth['Country']=df_country

df_gth = pd.concat([df_gth, X], axis=1, sort=False)
y = df_review.values
print(df_gth.head(5))
print(df_gth.shape)

def classify(clf,X_df, dem, color):
    a = np.zeros(shape=(6))
    px=[10,30,50,70,90]

    k=0
```

```
#spilt train-test data
for i in px:
    X_train, X_test, y_train, y_test = train_test_split(
        X_df, y, test_size = i/100)

    if(clf=='knn'):
        #KNN
        '''
        model = KNeighborsClassifier(n_neighbors=4)
        #Hyper Parameters Set
        params = {'n_neighbors':[4],
                  'leaf_size':[1,2,3,5],
                  'weights':['uniform', 'distance'],
                  'algorithm':['auto', 'ball_tree','kd_tree','brute'],
                  'n_jobs':[-1]}
        #Making models with hyper parameters sets
        knn = GridSearchCV(model, param_grid=params, n_jobs=1)
        '''
        knn = KNeighborsClassifier(algorithm= 'auto', leaf_size= 1, n_neighbors=4, weights = 'uniform')
        #Train the model using the training sets
        knn.fit(X_train, y_train)
        '''
        #The best hyper parameters set
        print("Best Hyper Parameters:\n",knn.best_params_)
        '''
```

```
        #Predict the response for test dataset
        y_pred = knn.predict(X_test)
    elif(clf=='nb'):
        #naive Bayes
        classifier = GaussianNB();
        classifier.fit(X_train, y_train)

        # predicting test set results
        y_pred = classifier.predict(X_test)
    elif(clf=='svm'):
        #SVM
        clf = SVC(C=2.5, gamma='auto', kernel='rbf')
        y_pred = clf.fit(X_train, y_train).predict(X_test)
    else:
        print('not valid classifier')

    # making the confusion matrix
    cm = confusion_matrix(y_test, y_pred)
    print(cm)
    accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy_score: ", accuracy)

    a[k]=accuracy
    k=k+1
#set plot features
```

```
    if(dem == 'original'):
        linestyle = 'solid'
    elif(dem == 'PCA'):
        linestyle = 'dotted'
    else:
        linestyle = 'dashed'
    plt.scatter(px,a[0:5])
    plt.plot(px,a[0:5],marker='',linestyle = linestyle, markersize=12, color=color, linewidth=4)



classify('knn',df_gth, 'original', 'blue')
classify('nb',df_gth, 'original', 'green')
classify('svm',df_gth, 'original', 'aqua')



#PCA
pca = PCA(n_components=2)
X_pca = pca.fit(df_gth).transform(df_gth)
# Percentage of variance explained for each components
print('Explained variance ratio (first two components): %s'
      % str(pca.explained_variance_ratio_))
classify('knn',X_pca, 'PCA','blue')
classify('nb',X_pca, 'PCA','green')
classify('svm',X_pca, 'PCA','aqua')
```

```
#LDA

lda = LinearDiscriminantAnalysis(n_components=2)

X_lda = lda.fit(df_gth, y).transform(df_gth)

classify('knn',X_lda, 'LDA', 'blue')

classify('nb',X_lda, 'LDA','green')

classify('svm',X_lda, 'LDA','aqua')


plt.show()
```