Machine Learning: Assignment 1

Luke Gibbons --- 15553883

ML Package Choice

After careful consideration on a number of machine learning packages I have decided to use Scikit-learn for the task at hand. The reason for choosing this package is primarily based on its ease of us, an important aspect as I have no prior knowledge on machine learning.

Compared to other machine learning packages I explored, Scikit-learn offers a high level library with an extensive range of easy to implement algorithms, including those for classification tasks. It also has extensive documentation on its API which includes tutorials along with descriptions on many core machine learning concepts. Finally, Scikit-learn has large community support that can be found across the web and is used my many companies, including the likes of Spotify.

Data Preparation

Preparing the dataset for input into Scikit-learn was a relatively straight forward process involving 4 main tasks.

1. **Load data**
   The Python package Pandas allows for the text file to be read and converted into a data frame - a 2-dimensional labeled data structure.

2. **Apply headings**
   The addition of headings make it easier to reference particular columns in the data frame.

3. **Drop *beer_id* column**
   The *beer_id* column is removed as it provides nothing useful in identifying a beers style.

4. **Define features and labels**
   The final step involves separating the features from the labels. In this case the label is the *style* column and the features are all the other columns in the data frame.

The Classification Algorithms

The two classification algorithms I decided to apply to the dataset are the k-nearest neighbors (k-NN) algorithm and the support vector machine algorithm (SVM). The scikit-learn algorithm flowchart (Scikit-learn developers 2020) helped me in deciding on the best algorithms for the job at hand.

K-NN

The k-nearest neighbors algorithm classifies data by the assumption that similar data exists in close proximity. It uses methods to find a number of training samples closest in distance to the new data point, it can then predict a label based on these 'neighbors'. A number of different techniques may be used to calculate the distance between points, the most popular being Euclidean distance (also known as straight line distance).

The value *k* represents the number of training samples (nearest neighbors) that are taken into consideration. The value of *k* is highly data dependent as a larger value will suppress the effects of noise but will also make the classification boundaries less evident.

SVM

Support Vector Machines aim to classify data by dividing the data points into clusters. It accomplishes this by drawing a hyperplane in N-dimensional space, where n is the number of features, that separates the data. Points on one side of the hyperplane can then be classified as Class A and those on the opposite side as Class B.

The classifier will attempt to draw a hyperplane that maximizes the distance between it (the hyperplane) and the data points. This helps to boost the classification confidence.

Results

The results for each algorithm are broken into three sections:

1. **Classification Accuracy**
   This is the simplest metric and is calculated as the number of correct predictions divided by the total number of predictions. It is given for both the training and test sets.

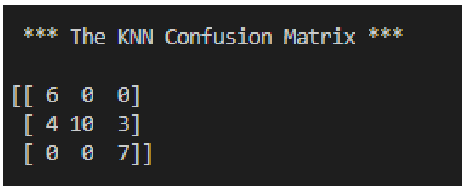
2. **Confusion Matrix**
   The diagonal elements of a confusion matrix represent the number of points for which the predicted label is equal to the true label while off-diagonal elements are

those that are mislabeled by the classifier. In this case, ale is represented by the first row and column, lager the second, and stout the third.

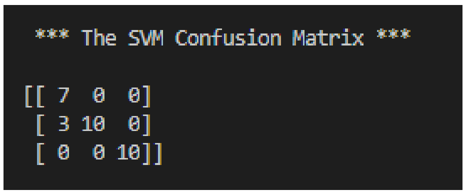
3. **Classification Report**
   The classification report provides is a built in scikit-learn metric for classification problems. It provides metrics such as recall (the number of Class A found divided by the total number of Class A in the dataset), precision (the percentage of examples labelled Class A that actually belong to the class), and f1-score (the average of precision and recall).

	K-NN		***SVM***	
Classification Accuracy	Test set: Training set:	77% 83%	Test set: Training set:	90% 98%

Confusion Matrix

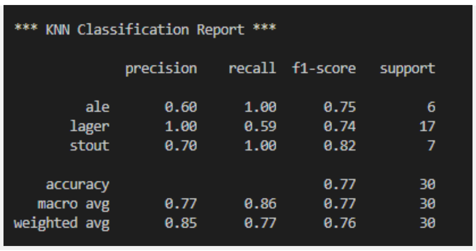


```
*** The KNN Confusion Matrix ***

[[ 6  0  0]
 [ 4 10  3]
 [ 0  0  7]]
```

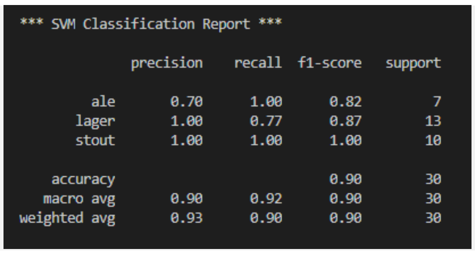


```
*** The SVM Confusion Matrix ***

[[ 7  0  0]
 [ 3 10  0]
 [ 0  0 10]]
```

Classification Report



```
*** KNN Classification Report ***

              precision    recall  f1-score   support

         ale       0.60      1.00      0.75         6
       lager       1.00      0.59      0.74        17
       stout       0.70      1.00      0.82         7

    accuracy                           0.77        30
   macro avg       0.77      0.86      0.77        30
weighted avg       0.85      0.77      0.76        30
```



```
*** SVM Classification Report ***

              precision    recall  f1-score   support

         ale       0.70      1.00      0.82         7
       lager       1.00      0.77      0.87        13
       stout       1.00      1.00      1.00        10

    accuracy                           0.90        30
   macro avg       0.90      0.92      0.90        30
weighted avg       0.93      0.90      0.90        30
```

Conclusion

After analysis of the results we can conclude that SVM performs better than the k-nn algorithm for this particular classification task.

K-nn performing worse than SVM may come down to a couple of reasons:

1. **Outliers and Bad Features**
   The k-nn algorithm is sensitive to both outliers and bad data, both of which may have been in the dataset. This is evident when failing to drop the *beer_id* column during the data preparation stage which results in a 40% classification accuracy on the test set (down 37%) using k-nn while SVM accuracy remains at 90%.

2. **High Dimensional Space**
   The k-nn algorithm is especially sensitive to a high dimensional space as points that are drawn from probability distribution don't ever tend to be close together. Our dataset has 8 dimensions (features to consider) which makes it difficult for k-nn to perform as well as the SVM algorithm.

References:

Lynn, S. (2019). *Using iloc, loc, & ix to select rows and columns in Pandas DataFrames* [Online]. Shane Lynn. Available at: https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/ (Accessed 30 October 2020)

Nelson, D. (2019). *Overview of Classification Methods in Python with Scikit-Learn* [Online]. Stack Abuse. Available at: https://stackabuse.com/overview-of-classification-methods-in-python-with-scikit-learn/ (Accessed 30 October 2020)

Scikit-learn developers. (2020). *Choosing the right estimator* [Online]. Scikit-learn. Available at: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html (Accessed 30 October 2020)

Scikit-learn developers. (2020). *Confusion Matrix* [Online]. Scikit-learn. Available at: https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html (Accessed 30 October 2020)

Scikit-learn developers. (2020). *Classification Report* [Online]. Scikit-learn. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html (Accessed 30 October 2020)

Scikit-learn developers. (2020). *Support Vector Machines* [Online]. Scikit-learn. Available at: https://scikit-learn.org/stable/modules/svm.html (Accessed 31 October 2020)

Scikit-learn developers. (2020). *Nearest Neighbors* [Online]. Scikit-learn. Available at: https://scikit-learn.org/stable/modules/neighbors.html (Accessed 31 October 2020)

Weinberger, K. (2018). *Lecture 2: k-nearest neighbors* [Online]. Cornell. Available at: https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html (Accessed 1 November 2020)