

**CSE523 Machine Learning**

**Weekly Report 6**

**Group Name: Precision Précis**

| **NAME** | **ENROLLMENT NUMBER** |
| --- | --- |
| Twinkle Popat | AU2040216 |
| Kathan Bhavsar | AU2040170 |
| Rushali Moteria | AU2040210 |
| Neel Buddhdev | AU2040176 |

1. **Tasks performed in the week and outcomes of the tasks performed.**
2. **Using DataFrame to train and test**

We take the Count Vectoriser array (cv\_arr1), Binary array (binArray1) and Sentence Length array (sen\_lens1) and convert these arrays in list form.

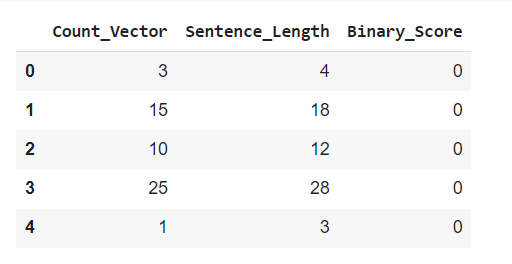
| df\_cv = [] df\_san = [] df\_bin = []  for cv in cv\_arr1:  for val in cv:  df\_cv.append(val)  for art in sen\_lens1:  for sent in art:  df\_san.append(sent)  for bin in binArray1:  for b in bin:  df\_bin.append(b) |
| --- |

Then we use DataFrame() function from Pandas Library to create a DataFrame (df) from the three lists.

| df = pd.DataFrame(list(zip(df\_cv,df\_san,df\_bin)), columns = ['Count\_Vector','Sentence\_Length','Binary\_Score']) |
| --- |

**The resulting Data Frame is as shown below:**

The output in the output of the head of the dataframe. It contains three columns 'Count\_Vector', 'Sentence\_Length', and 'Binary\_Score'. Each column in the dataframe corresponds to the values from the df\_cv, df\_san, and df\_bin lists respectively. We convert the arrays to dataframes in order to easily perform several tasks such plotting a graph.



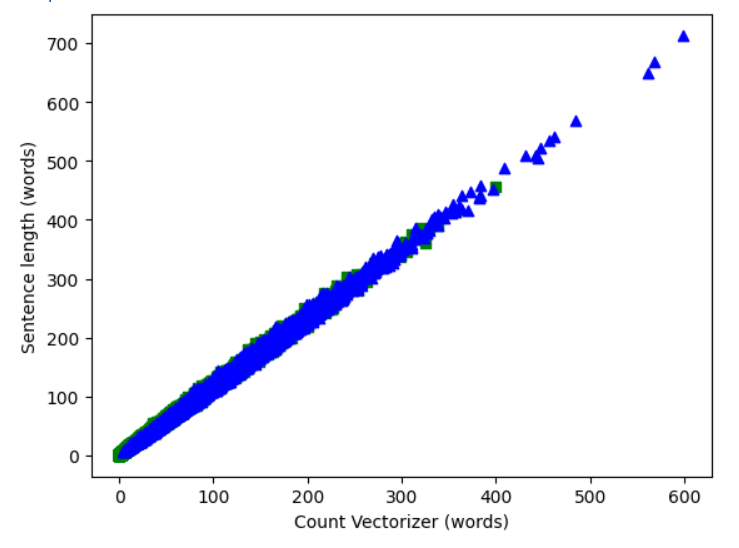
1. **Ploting the graph for Count Vectoriser versus Sentence Length**

We divide the data into two classes, namely Class 1 (important sentence) and Class 0 (non-importnant sentence). We create two dataframes, df0 and df1 by filtering the values of Binnary Score in the original dataframe (df). All the tuples having Binnary\_Score as ‘0’ are stored in df0 (representing class 0) and the ones with score ‘1’ are stored in df1 (representing class 1). Using these two dataframes we plot the the values of count vectoriser and the sentence length for both the classes.

| df0 = df[df.Binary\_Score == 0] # Dataframe for class 0 df1 = df[df.Binary\_Score == 1] # Dataframe for class 1  plt.xlabel('Count Vectorizer (words)') plt.ylabel('Sentence length (words)') plt.scatter(df0['Count\_Vector'],df0['Sentence\_Length'],color='green',marker='s) plt.scatter(df1['Count\_Vector'],df1['Sentence\_Length'],color='blue',marker='^') |
| --- |

**The below in the output of the plot:**

The blue represents class 1 and green represents class 0.

****

**Inference from the graph:**

In the graph, we observe that the green and blue labels overlap each other. This shows that count vectorization would not be an efficient method to distinguish between the two classes i.e. Class 1 (important sentence) and Class 0 (non-important sentence). This is because two sentences may be in different classes, but at the same time, they have the same score my count vectoriser method. To overcome this problem, we will user NER with count vectoriser and set thresholds to differentiate between the two classes.

1. **Testing different kernels**

We are training the SVM classifier with different kernels - Linear, Polynomial, Gaussian RBF, and sigmoid using the sci-kit-learn library. And then, we are using the ‘score’ method to evaluate the accuracy of the trained classifier on the test set. The resulting accuracy is then printed.

Inititally we take the first two columns (which represent features) of the dataframe as x input and the Binnary Score as y input. Then we divide the data into test data (20%) and train data (80%).

| x = df.drop(['Binary\_Score'],axis='columns') y = df.Binary\_Score x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size =0.2) |
| --- |

1. **Linear Kernel**

This kernel tries to fit the dataset on a straight line.

| model\_lin = SVC(kernel='linear') model\_lin.fit(x\_train,y\_train) D = model\_lin.score(x\_test ,y\_test) Score\_arr.append(D) print(D) |
| --- |

**Output:**

****

1. **Polynomial Kernel**

This kernel function can classify data that is not linearly separable. It uses the polynomial function to map the data into a higher dimensional space.

Here, the degree parameter is set to 3 which means that the classifier uses a polynomial kernel function of degree 3 for classification.

| model\_poly = SVC(kernel='poly', degree=3) model\_poly.fit(x\_train,y\_train)  D = model\_poly.score(x\_test,y\_test)  Score\_arr.append(D)  print(D) |
| --- |

**Output:**

****

1. **Gaussian RBF Kernel**

This kernel function is also known as the Gaussian kernel. It is useful when the data is not linearly separable and can create a non-linear decision boundary.

| model\_lin = SVC(kernel='rbf') model\_lin.fit(x\_train,y\_train) D = model\_lin.score(x\_test ,y\_test) Score\_arr.append(D) print(D) |
| --- |

**Output:**

****

1. **Sigmoid Kernel**

This kernel function is useful or non-linear data. It transforms the data into a higher-dimensional space using a sigmoid function.

| model\_lin = SVC(kernel='sigmoid') model\_lin.fit(x\_train,y\_train) D = model\_lin.score(x\_test ,y\_test) Score\_arr.append(D) print(D) |
| --- |

**Output:**

****

**Observations:**

We have observed that Linear kernel took more time (about 30 mins) than the kernel kernels because the input data in our case is highly random and it does not exhibit any linear relationship. Since the linear kernel is unable to capture the non-linear relationships between the text features, it requires more iterations to converge to an optimal solution. While on the other hand, the polynomial kernel easily captures the non-linear relationships and faster convergence and fitting of the data.

1. **Tasks to be performed in the upcoming week.**

In the upcoming week we will try to merge NER concept with Count vectoriser. Instead of just counting the words in the sentence, we will count the number of entity words in each sentence. If the sentence has no entity word then it will have a score of ‘0’. Then we will count the total number of entity words in the entire article and give relative scores of importance (Number of entity words in each sentence / The number of entity words in the article). Then we will set a threshold score, if the sentence has score higher than this score, it will belong to Class 1, otherwise it will belong to Class 0.