# **HW 1 (584-Rangwala): Movie Review Classification**

# Pankaj Kumar Jatav – G10338769 ID – pjatav miner – criminal.

**Goal:**

Implement the Nearest Neighbor Classification Algorithm to predict movie rating files.

**Approach:**

I follow the below diagram step to solve this assignment.



**Preprocessing of Data:**

Below are the steps taken to preprocess that data.

1. Remove HTML tag
2. Tokenize using WhitespaceTokenizer
3. Remove English stop words
4. Remove no dictionary words
5. Lemmatize and Stemming(SnowballStemmer)

doc = cleanhtml(doc)  
tk\_data = tk.tokenize(doc)  
list\_data = [w for w in tk\_data if w.lower() not in stopwords.words('english')]  
list\_data = [w.lower() for w in list\_data if w.lower() in words and w.isalpha()]  
lem\_data = lemmatize\_sentence(list\_data)  
data = [self.stemmer.stem(w) for w in self.analyzer(lem\_data)]

**Model Training:**

Below are the steps taken to create model from training data.

1. Create the features from preprocessed data which was around 17K in my case.
2. Use TF-IDF approach to train model (17K x training set length matrix) each records in training set.

I used the Sklearn TfidfVectorizer text feature extraction python module to perform this task.

vectorizer = TfidfVectorizer()  
vec = vectorizer.fit(train\_data)  
train\_vec = vec.transform(train\_data)  
test\_vec = vec.transform(test\_data)

**KNN:**

To calculate the nearest neighbors,

1. I have cross multiple the train matrix with test matrix.
2. Calculate distance with all the reviews for all the test data.
3. Choose the N nearest neighbors.
4. Sum the N nearest neighbors rating.
5. If sum of rating is positive, then predict as +1 and if sum is negative, review predict as -1.

Below code will return array of ratings of prediction for all the test reviews

def find\_knn(row):  
 score = 0  
 for w in row.argsort()[-221:][::-1]:  
 score += train\_class[w]  
 if (score > 0):  
 return 1  
 else:  
 return -1  
  
test\_vec = test\_vec.transpose()  
value = np.matmul(training\_vec.toarray(), test\_vec.toarray())  
save\_plot(value, 'knn.png')  
predict = np.apply\_along\_axis(find\_knn, axis=0, arr=value)

**Cross Validation:**

To Cross validate the model accuracy. I split the document in 4 diff ratio (0.66, 0.34), (0.7, 0.3), (0.8, 0.2), (0.9, 0.1). And getting the highest accuracy which is 80.4% in (0.8, 0.2) ratio.

Below function used to visualize the prediction data.

print(confusion\_matrix(train\_class, prediction))  
print(accuracy\_score(train\_class, prediction))

Test Validation:

Save the KNN prediction to out.dat file and upload to miner for accuracy.

np.savetxt('./data/out.dat', predict, delimiter=',', fmt='%i')

**Step To Run the program:**

Run main.py file

>>python3 main.py

It will give 5 options as bellow. Please always run 1st option processed\_train\_data.csv and processed\_test\_data.csv are not present in data folder.

Please select a options

1. Prepossessed data set

2. Perform cross validation on training data set

3. Perform data prediction on test set

4. Above all in sequence

5. Exit

Option 1. Allow user to pre-processed train and test data and save those file into csv files.

Option 2. It will allow do cross validation on different model selection as bellow. Once you choose the model it will train model and predict the output for test data and print confusion matrix and accuracy score.

Please choose the model selection

1. 66 Training Set - 34 Test Set

2. 70 Training Set - 30 Test Set

3. 80 Training Set - 20 Test Set

4. 90 Training Set - 10 Test Set

Enter your choice:

Option 3. It allows use to predict rating for test data. And it will save file into data folder as out.dat.

Option 4. It combines the option 1 to 3.

Option 5. To terminate the prompt.

**KNN calculation:**

Let say we have X features in train data which we got from data processing. And we have 25K reviews in test and train data.

Dimension:

TrainVector (25000, X )

TestVector (25000, X )

Now, Multiple both matrixes

DisanceMatrix = TrainVector \* Transpose(TestVector)

Disance\_Matrix will have 25K x 25K matrix which will show the distance between all the TrainVector, TestVector. It will increase performance as we do not have to iterate through each review.

Select the k-nearest neigher for each TestVection from DistanceMatrix which is defined as find\_knn function in code.

**K-Value Selection:**

To select the value of K, loop the K-value from 1 to 250 and selected the one who was having higher accuracy. In my case It was 221. In my case It was 221. Refer perform\_knn\_values function.

**Plots:**

Below are matrixes graph for cross validation.

1. Matrix plot for mode. Colors show Term Frequency for each features and review.

Chart, diagram

Description automatically generated

1. Matrix plot for test data.

A picture containing text, appliance, stove, kitchen appliance

Description automatically generated

1. Matrix plot for product of model and test data to calculate KNN (5000 x 20000 matrix).

Timeline

Description automatically generated

Below are matrixes graph for predict for test data.

1. Matrix plot for model (25000 x ~16K matrix). Colors show Term Frequency for each features and reviews.

Chart

Description automatically generated

1. Matrix plot for test data (25000 x ~16K matrix).

Chart

Description automatically generated

1. Matrix plot for product of model and test data to calculate KNN (25000 x 25000 matrix).

Chart

Description automatically generated