CS777 – Week 5 Homework Submission Template

**!!!! PLEASE RENAME THIS DOCUMENT WITH YOUR NAME AND LASTNAME !!!!**

**Task 1 – Vectorize the Data**

* Vectorize the text documents using the available functions in the MLlib

|  |
| --- |
| *start\_time = time.time()*    *tokenizer = Tokenizer(inputCol="text", outputCol="words")*  *wordsData = tokenizer.transform(df)*    *remover = StopWordsRemover(inputCol="words", outputCol="filtered\_words")*  *wordsData = remover.transform(wordsData)*    *cv = CountVectorizer(inputCol="filtered\_words", outputCol="rawFeatures", vocabSize=5000)*  *cv\_model = cv.fit(wordsData)*  *featurizedData = cv\_model.transform(wordsData)*    *idf = IDF(inputCol="rawFeatures", outputCol="features")*  *idfModel = idf.fit(featurizedData)*  *rescaledData = idfModel.transform(featurizedData)*    *rescaledData.cache()*    *# Subtasks*  *vocab = cv\_model.vocabulary*  *print("First 10 words in the vocabulary : ", vocab[:10])*    *end\_time = time.time()*  *print("Total time to vectorize the data: {:.2f} seconds".format(end\_time - start\_time))*  *print("-----------------------------------------------------------------------------------")*  *print()* |
| *start\_time\_test = time.time()*    *wordsData\_test = tokenizer.transform(test)*  *wordsData\_test = remover.transform(wordsData\_test)*  *featurizedData\_test = cv\_model.transform(wordsData\_test) # Use the same CountVectorizerModel fitted on TRAIN data*  *rescaledData\_test = idfModel.transform(featurizedData\_test) # Use the same IDFModel fitted on TRAIN data*    *rescaledData\_test.cache()*  *print("The vocabulary in TRAIN is reused in TEST, so the First 10 words in the vocabulary : ", vocab[:10])*  *end\_time\_test = time.time()*  *print("Total time to vectorize the TEST data: {:.2f} seconds".format(end\_time\_test - start\_time\_test))*  *print("-----------------------------------------------------------------------------------")*  *print("==============================================================================================================================")*  *print()* |

* Print the first ten words of the vocabulary

|  |
| --- |
| *First 10 words in the vocabulary : ['also', 'first', 'one', 'new', 'two', 'may', 'made', 'time', 'many', 'three']* |
| *The vocabulary in TRAIN is reused in TEST, so the First 10 words in the vocabulary : ['also', 'first', 'one', 'new', 'two', 'may', 'made', 'time', 'many', 'three']* |

* Print the total time needed to vectorize the data

|  |
| --- |
| *Total time to vectorize the data: 279.71 seconds* |
| *Total time to vectorize the TEST data: 0.31 seconds* |

**Task 2 – Using the Logistic Regression Model**

* Print the F1-measure and **confusion matrix**

|  |
| --- |
| *F1: 0.9868421052631579*  *Confusion Matrix:*  *TP: 375 FN: 2*  *FP: 8 TN: 18339* |

* Print the total time needed to train the model

|  |
| --- |
| *The total time needed to train the model: 143.53 secs* |

* Calculate the performance metrics

|  |
| --- |
| *Performance Metrics: Logistic Regression*  *Precision: 0.97911227154047*  *Recall: 0.9946949602122016*  *F1: 0.9868421052631579*  *Confusion Matrix:*  *TP: 375 FN: 2*  *FP: 8 TN: 18339* |

**Task 3 – Using the SVM Model**

* Print the F1-measure and **confusion matrix**

|  |
| --- |
| *F1: 0.9946949602122016*  *Confusion Matrix:*  *TP: 375 FN: 2*  *FP: 2 TN: 18345* |

* Print the total time needed to train the model

|  |
| --- |
| *The total time needed to train the model: 30.16 secs* |

* Calculate the performance metrics

|  |
| --- |
| *Performance Metrics: SVM*  *Precision: 0.9946949602122016*  *Recall: 0.9946949602122016*  *F1: 0.9946949602122016*  *Confusion Matrix:*  *TP: 375 FN: 2*  *FP: 2 TN: 18345* |

**Task 4 – Feature Selection**

* Describe your feature selection technique and discuss why you chose it

|  |
| --- |
| **Description:**  ***CountVectorizer:***  *Purpose: Converts a collection of text documents to a matrix of token counts.*  *Working: It tokenizes the text documents and counts the occurrences of tokens in the data. The result is a sparse matrix representation of the documents.*  ***TF-IDF (Term Frequency-Inverse Document Frequency):***  *Purpose: Evaluate how important a word is to a document in a collection of documents.*  *Working: It considers both the frequency of a word in a document and the frequency of the word across all documents in the corpus.*  ***Reasons for Choosing CountVectorizer and TF-IDF:***  ***Handling Text Data:***  *Both techniques are adept at handling text data and converting it into a format that can be fed into machine learning models.*  ***Dimensionality Reduction:***  *TF-IDF can potentially reduce dimensions by assigning lower weights to less informative words, thereby allowing models to focus on more important terms.*  ***Handling Different Vocabulary:***  *CountVectorizer helps in handling different vocabularies in text data by creating a consistent feature space for model training.*  ***Highlighting Informative Words:***  *TF-IDF emphasizes words that are more unique and informative to the document, providing a normalized way to consider word importance.*  ***Improving Model Performance:***  *Both techniques can enhance model performance in NLP tasks by providing numerical and normalized representations of text data.* |

* Discuss if your dimension reduction approach is applicable to very large data sets

|  |
| --- |
| ***Applicability to Large Datasets:***  ***Scalability:***  *Both CountVectorizer and TF-IDF are scalable and can be applied to large text datasets. They generate sparse matrices, which are memory-efficient, especially for large and high-dimensional data.*  ***Memory Efficiency:***  *The sparse matrix representation used by both techniques ensures that memory usage is optimized, making them suitable for large datasets.*  ***Computational Efficiency:***  *While both techniques are computationally efficient for moderate-sized data, for extremely large datasets, considerations regarding computational resources and time need to be taken into account.*  ***Online Learning:***  *Adapting these techniques for online learning scenarios, where data is continuously generated, may require utilizing incremental or partial-fit versions of models and vectorizers.*  ***Additional Consideration for Dimensionality Reduction:***  *While CountVectorizer and TF-IDF provide numerical representations of text data, further dimensionality reduction (from 5K to 200) might require additional techniques like PCA, Truncated SVD, or feature selection methods. Ensuring that the reduced feature space retains crucial information is vital for maintaining model performance and interpretability.*  ***Conclusion:***  *CountVectorizer and TF-IDF are widely used for converting text data into numerical format and can be applied to large datasets due to their use of sparse matrix representations. However, when dealing with very large datasets and significant dimensionality reduction, additional techniques and considerations regarding computational efficiency and data representativeness must be taken into account to ensure robust model training and prediction.* |

* Repeat tasks 2 and 3 with reduced features and report the F1 measure, confusion matrix, and computation time

|  |
| --- |
| ---------------------------------- Logistic Regression ----------------------------------  True Positives: 346  True Negatives: 18328  False Positives: 19  False Negatives: 31  accuracy: 99.73296304208502 %  Performance Metrics: Logistic Regression  Precision: 0.947945205479452  Recall: 0.9177718832891246  F1: 0.9326145552560647  Confusion Matrix:  TP: 346 FN: 31  FP: 19 TN: 18328  The total time needed to train the model: 122.72 secs  Evaluate the model: 0.19 secs  Test the model: 55.99 secs  Total Time: 178.90 secs |

|  |
| --- |
| *-------------------------------------- SVM model --------------------------------------*  *True Positives: 347*  *True Negatives: 18335*  *False Positives: 12*  *False Negatives: 30*  *accuracy: 99.77568895535141 %*  *Performance Metrics: SVM*  *Precision: 0.9665738161559888*  *Recall: 0.9204244031830239*  *F1: 0.9429347826086956*  *Confusion Matrix:*  *TP: 347 FN: 30*  *FP: 12 TN: 18335*  *The total time needed to train the model: 28.28 secs*  *Evaluate the model: 0.07 secs*  *Test the model: 3.50 secs*  *Total Time: 31.86 secs* |

* Comment on the differences in the results

|  |
| --- |
| *For the difference, the svm model is more accuracy, because it has higher accuracy, higher precision, higher recall and higher F1 score, for the cost time, SVM has shorter time* |

**Spark History Output:**

**Task 1:**

|  |
| --- |
| *spark-history-task1-3* |

**Task 2:**

|  |
| --- |
| *spark-history-task1-3* |

**Task 3:**

|  |
| --- |
| *spark-history-task1-3* |

**Task 3:**

|  |
| --- |
| spark-history-task4 |