DWDM PROJECT

All Roll Numbers of your Team:	**18BCD7008** **18BCD7080**
Student names in your team :	**Kathal Aditya Rajendra** **Anshika Singh**
Contribution by each team member in percentage to complete this project:	**Kathal Aditya Rajendra : 51%** **Anshika Singh : 49%**
Slot (L1/L2/L4):	**L1**
Title of the Project:	**Daily News for Stock Market Prediction**
Objective of the Project (What exactly the project is about?)	**The goal of the project is to predict whether the stock goes up or down based on top 25 headlines.**

Dataset Link	**Number of rows and Columns**	**About columns**
https://www.kaggle.com/aaron7sun/stocknews?select=upload_DJIA_table.csv	**Rows: 1989** **Columns: 32**	**Number of Categorical columns: 1** **Number of Integer/Float Columns: 6** **Number of Pure String Columns: 25**
		Unique Values in each Column: 1. **Open: 1980** 2. **High:1983** 3. **Low:1980** 4. **Close:1978** 5. **Volume:1897** 6. **Adj Close:1978** **In string column,every text is unique.**

Challenges identified in the project	**How did you address that challenge?**	**References**
1) The dataset doesn't contain any NAN values.	**By creating NAN values randomly in different chosen columns with the help of random package.We haven't generated random NAN values in Date and Label columns**	https://stackoverflow.com/questions/36413314/filling-missing-data-by-random-choosing-from-non-missing-values-in-pandas-datafr

2) Input contains NAN,infinity or a value too large for dtype ('float32')-dealing with nan values	**By either dropping NAN values from that column if not required or replacing NAN values with different methods.**	https://datascience.stackexchange.com/questions/11928/valueerror-input-contains-nan-infinity-or-a-value-too-large-for-dtypefloat32
3) Applying new models	**By importing respective models after reading from the documentation page.**	
4) Dealing with string columns	**For string columns we went through various materials present online and chose different techniques.**	https://www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-data-predictive-python/
5) Improving accuracy	**When using the uni-gram approach for string columns specifically the maximum accuracy achieved was 57% thus we had to look into different techniques and hyper parameters to increase that.**	https://towardsdatascience.com/available-hyperparameter-optimization-techniques-dc60fb836264

Without Pre-processing- Different Algorithms	**Performance(Accuracy/other confusion matrix measures)**	**Which model worked well on the test data and WHY?**
Light GBM	**Accuracy: 0.87 Recall: 0.86**	**Gradient Boosting(n_estimators=200) worked better than other models because overall it's accuracy was high with good recall value.**
Stratified 10 Fold (base = RandomForest)	**Accuracy: 0.81 Recall: 0.85**	
Stratified 10 Fold (base =XGBoost)	**Accuracy: 0.82 Recall: 0.84**	
Logistics Regression	**Accuracy: 0.51 Recall: 1**	
K – Nearest Neighbours(N = 5)	**Accuracy: 0.61 Recall: 0.67**	
K – Nearest Neighbours(N = 11)	**Accuracy: 0.58 Recall: 0.69**	
K – Nearest Neighbours(N = 21)	**Accuracy: 0.56 Recall: 0.69**	

Decision Tree (criterion = 'gini' , max_dept = 10)	**Accuracy: 0.58** **Recall: 0.83**	
Decision Tree (criterion = 'entropy' , max_dept = 10)	**Accuracy: 0.6** **Recall: 0.82**	
Random Forest(n_estimators=10)	**Accuracy: 0.82** **Recall: 0.79**	
Random Forest(n_estimators=100)	**Accuracy:0.84** **Recall:0.87**	
Random Forest(n_estimators=200)	**Accuracy:0.85** **Recall:0.90**	
Gradient Boosting(n_estimators=10)	**Accuracy:0.72** **Recall:0.91**	
Gradient Boosting(n_estimators=100)	**Accuracy:0.86** **Recall:0.90**	
Gradient Boosting(n_estimators=200)	**Accuracy: 0.88** **Recall: 0.90**	
XGBoost	**Accuracy: 0.83** **Recall: 0.85**	
Meta Classifier Stacking **Base classifier: Logistics Regression** **Estimators: KNN** **Decision Tree** **GradientBoosting**	**Accuracy: 0.37** **Recall: 0.68**	

Meta Classifier Stacking **Base classifier: Logistics Regression** **Estimators: Random forest** **XGBClassifier**	**Accuracy: 0.51** **Recall: 1**	

Without Preprocessing(String Columns only)

Without Pre-processing - Different Algorithms	Performance(Accuracy/ other confusion matrix measures)	Which model worked well on the test data and WHY?
Logistics Regression	0.50	**No algorithm performed better. Relatively,Light GBM stands out better.**
KNN	0.55	
Naive Bayes	0.50	
Decision Tree	0.55	
Random Forest	0.49	
Gradient Boosting	0.53	
XGB	0.53	
LGB	0.56	
StackingClassifier(RF ,XGBoost)	0.55	
Stratified 10 Fold(base=Random Forest)	0.53	

ID	Which Pre-processing technique you applied?	Why you applied that pre-processing Technique?	References
1	**Convert all string to lowercase**	**String matching is usually case sensitive and thus has the same words but may contain characters in different cases(upper , lower) thus forming a new column in the document term matrix. This increases the dimensionality of the dataset.**	https://www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-data-predictive-python/
2	**Remove punctuations**	**Punctuations are present in every text. These do not provide any value to the model when we use different algorithms. They serve as noise.**	https://www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-data-predictive-python/
3	**Remove Stop Words**	**Stop Words are words that are used commonly in a language. Thus they have very high frequency but have no value to the model so we remove them.**	https://www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-data-predictive-python/
4	**Remove Numerical Values**	**This depends on the objective of the project to remove numerical values or not. This project did not require them se we remove all numerical values.**	https://www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-data-predictive-python/
5	**Porter Stemmer**	**The most used stemming algorithm . It is very gentle while creating stems when compared to lancaster.**	https://www.geeksforgeeks.org/python-stemming-words-with-nltk/
6	**Snowball Stemmer**	**This algorithm is also known as the Porter2 stemming algorithm. It is almost universally accepted as better than the Porter stemmer, even being acknowledged as such**	https://www.geeksforgeeks.org/snowball-stemmer-nlp/

		by the individual who created the Porter stemmer	
7	**Lancaster Stemmer**	**This one is the most aggressive stemming algorithm of the bunch.**	https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8
8	**Lemmatization**	**Lemmatization converts the word into its root word, rather than just stripping the suffices. Thus reducing the number of columns.**	https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8
9	**Removal Of Words based on Document Frequency**	**The intuition behind document frequency is that a word is not of much use to us if it's appearing in all the documents.**	https://www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-data-predictive-python/
10	**N – Gram**	**N-Gram analysis helps us in understanding word association and reduce the dimensionality of the dataset.**	https://www.tidytextmining.com/ngrams.html
11	**Replacing NAN values with mean and median.**	**We can try to acquire better accuracy.It also decreases the loss of data.**	Class Material
12	**Replacing NAN values with forward fill and backward fill.**	**It is one more technique in which we can fill NAN values.Forward fill**	Class Material
13	**KNN Imputer**	**To complete missing values using K nearest neigbours.**	Class Material
14	**Standardization and Normalization of columns**	**Standardized values are useful to track data which is not easy to compare otherwise.Moreover,data normalization gets rid of a number of anomalies.**	Class Material

15	**Dealing with Outliers**	**Outliers may lead us to wrong results,so it's better if we remove them from our dataset.**	Class Material
16	**Smote Analysis**	**To address imbalanced datasets which is to oversample the minority class.**	https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

UNI GRAM

Pre-processing technique name?	Data Mining Algorithms you applied?	Performance(Accuracy/other confusion matrix measures) (Before pre-processing)	(After Pre-processing)	Which model worked well on the test data and WHY?
Drop Na + toLowerCase	KNN(N = 5)	Accuracy: 0.61 Recall: 0.67	Accuracy:0.62 Recall:0.66	**Light GBM model worked well on the test data since it's based on decision tree,it splits the tree leaf with the est fit whereas other boosting algorithms split the leaf tree depth wise.Hence,its faster and efficient. Moreover,boosting always aims at reducing bias in the dataset.That's why LGM performs much better.**
KNN Imputer+toLowerCase	LGM	Accuracy: 0.87 Recall: 0.86	Accuracy:0.89 Recall: 0.90	
Fill Na with mean + toLower + remove numericals	XG Boost	Accuracy:0.83 Recall:0.85	Accuracy:0.84 Recall:0.88	
Fill Na with mean + remove Stop words + Lancaster Stemming	XG Boost	Accuracy:0.83 Recall:0.85	Accuracy:0.86 Recall: 0.94	
KNN Imputer + remove numerical + lemmatization	LGM	Accuracy:0.87 Recall:0.86	Accuracy: 0.90 Recall: 0.92	

Fill forward + tokenization(min_df = 0.01 , max_df = 0.95)	Gradient Boosting	Accuracy:0.85 Recall:0.90	Accuracy:0.90 Recall:0.92	
Fill backward + remove stop words + remove numerical+lemmatization + token(min_df = 0.1 , max_df = 0.85)	LGM	Accuracy:0.87 Recall:0.86	Accuracy:0.88 Recall:0.90	

UNI Gram (String Columns only)

Pre-processing technique name?	**Data Mining Algorithms you applied?**	**Performance(Accuracy/other confusion matrix measures)**		**Which model worked well on the test data and WHY?**
		(Before pre-processing)	**A(After pre-processing**	
toLowerCase+remove stop words+remove numerical+remove punctuation+snowball stemmer+tokenization(min_df=0.01,max_df=0.95)	1.Logistic Regression 2.KNN 3.Naive Bayes 4.Decision Tree 5.Random Forest 6.Gradient Boosting 7.XGB 8.LGB 9.StackingClassifier(RF,XGBoost)	0.50 0.55 0.50 0.55 0.49 0.53 0.53 0.56 0.55 0.53	0.49 0.53 0.49 0.56 0.53 0.51 0.51 0.52 0.56 0.53	**No algorithm performed better. Relatively,Decision tree stands out better.**

	10.Stratified 10 Fold(base=Random Forest)			

BI GRAM(String Columns only)

When we tried to run bi gram analysis with both numerical and string columns the google colab notebook re started throwing an error that RAM usage was exceeding the limit. The highest accuracy for uni gram analysis for only string columns was 57%. Here we can see that the highest is above 80.

Pre-processing technique name?	**Data Mining Algorithms you applied?**	**Performance(Accuracy/other confusion matrix measures)** **(Before pre-processing)**	**(After Pre-processing)**	**Which model worked well on the test data and WHY?**
toLowerCase	Stratified 10 Fold (base = RandomForest)	Accuracy: 0.81 Recall: 0.98	Accuracy: 0.82 Recall: 0.98	**Stratified 10 fold (base=Random Forest)model works better than other models because it**
toLower + remove numericals	Decision Tree (criterion = 'gini' , max_dept = 10)	Accuracy: 0.55 Recall: 0.96	Accuracy: 0.56 Recall: 0.95	

				trains on stratified data again and again which creates a bias in it.Due to this,we are getting higher accuracy and higher recall value.Moreover,thi s also handles noise in the dataset.
remove Stop words + Lancaster Stemming	KNN(N = 5)	Accuracy: 0.50 Recall: 1	Accuracy: 0.51 Recall: 1	
remove numerical + lemmatization	Gradient Boosting(n_estimators=200)	Accuracy: 0.83 Recall: 0.86	Accuracy: 0.81 Recall: 0.88	
toLowercase + remove stop words + remove numerical + Remove punctuation + porter stemmer + tokenization(min_df = 0.01 , max_df = 0.95)	Meta Classifier Stacking Base classifier: Light GBM Estimators: KNN Decision Tree GradientBoosting	Accuracy: 0.47 Recall:0.52	Accuracy: 0.58 Recall: 0.63	
toLowercase + remove stop words + remove numerical+lemmatization + token(min_df = 0.1 , max_df = 0.90) + max_number = 700	XGBoost	Accuracy:0.75 Recall: 0.91	Accuracy: 0.84 Recall: 0.91	
toLowercase + remove stop words + remove numerical + Remove punctuation + snowball stemmer + tokenization(min_df = 0.01 , max_df = 0.95)	RandomForest(n = 200)	Accuracy: 0.84 Recall: 0.97	Accuracy: 0.87 Recall: 0.82	

Summary:

Number of Pre-processing Techniques applied with their names: **19**

1. Replacing Nan values with mean
2. Replacing Nan values with median
3. forward fill
4. backward fill
5. KNN Imputer
6. dropping NAN values
7. Standard Scaler(Standardization)

8. Smote Analysis
9. Convert all string to lowercase
10. Remove punctuations
11. Remove Stop Words
12. Remove Numerical Values
13. Stemmer
14. Lemmatization
15. Removal Of Words based on Document Frequency,
16. N – Gram.
17. Removing Outlier using Z Score
18. Removing outliers using IQR
19. Removing outliers using quantiles

Number of Data Mining Algorithms applied with their names: 10

1. K-Nearest Neighbour(KNN)
2. Logistic Regression
3. Random Forest Classifier
4. Decision Tree Classifier
5. Gradient Boosting
6. Gaussian Naive Bayes
7. XG Boost
8. Light GBM
9. Stratified 10 Fold (base = RandomForest)
10. Stratified 10 Fold (base = XGBoost)

Which algorithm showed highest performance after "All" pre-processing techniques and WHY?:

In UNIGRAM analysis,LGM showed the highest performance with an accuracy of 90% and has low number of false positives and false negatives when compared to other models.Moreover,we know that boosting works while training and testing the missing values constantly,which is one of the reasons for it's higher accuracy.
In BIGRAM analysis,Stratified Fold 10(base=Random Forest) works better than other models because it trains on stratified data again and again which creates a bias in it.Due to this,we are getting higher accuracy and higher recall value.Moreover,this also handles noise in the dataset.

Conclusion-Write in your own words:

In conclusion achieved an accuracy above 80 percent in both cases when considering only string columns and string + numerical. We saw drastic changes in recall and accuracy when hyper parameters are changed and association of words is considered. When training the model.

Since we had limited rows and there was a bias in result in unigram analysis of only strings columns we used Stratified k fold technique to deal with this and also looked at various n-gram approaches.

We also applied a better version of XGBoost that is Light GBM which performed better than it in almost all cases.