**IT2386 Text and Social ANALYTICS PROJECT**

**Model Evaluation report**

|  |  |
| --- | --- |
| **Submitted By:** | Gangula Karthik (223715Y)**\*** |
|  | Ng Jun Ming (220080D) |
|  | Seah Pin Shien (220273H) |
| **Team Name:** | Dip Sum |
| **Tutor:** | Ms Jane Zhang  Mr Lim Sing Tat |
| **Submission:** | 11 February 2024 |

Table of Contents

[Executive Summary 3](#_Toc158532035)

[Introduction 3](#_Toc158532036)

[Methodology 3](#_Toc158532037)

[Data Utilization 3](#_Toc158532038)

[Model Development 4](#_Toc158532039)

[Results 6](#_Toc158532040)

[Consolidated Test Data outcomes 6](#_Toc158532041)

[Model Outcome Comparison 7](#_Toc158532042)

[**Karthik** 7](#_Toc158532043)

[**Jun Ming** 7](#_Toc158532044)

[**Pin Shien** 7](#_Toc158532045)

[Discussion 7](#_Toc158532046)

[Conclusion 7](#_Toc158532047)

# **Executive Summary**

The project object is to develop a multi class text classifier that can classify whether the review about a particular airline is coming from either a promoter, detractor, or a passive. To do this the text data has been web scraped from various sources on the internet, preprocessed, analyzed, modelled, and finally evaluated for its accuracy and relevance in solving the given problem. This report will dive deeper into the model performance to assess its suitability in solving the given problem.

# **Introduction**

In the realm of customer experience management, Net Promoter Score (NPS) measures the likelihood of customers recommending a company's products or services to others, thereby reflecting their overall satisfaction and loyalty levels. This metric holds immense importance for businesses as it directly correlates with growth, profitability, and long-term success.

A high NPS indicates a strong base of loyal customers who are likely to advocate for the brand, leading to increased customer retention, positive word-of-mouth (WOM) marketing, and ultimately, sustainable business growth. Conversely, a low NPS signals areas of improvement in customer experience and highlights potential issues that need addressing to prevent customer churn and negative publicity.

Airline companies was looked at particularly as they operate in a fiercely competitive environment where customer experience plays a pivotal role in shaping brand perception and influencing purchasing decisions. By analyzing airline reviews the project seeks to achieve the following goals:

# **Methodology**

## Data Utilization

The group used the same methods to:

* Standardize the target column by converting from ratings into NPS\_category
* Standardize the date column from different datetime formats into a single uniform format.

A brief overview of how each person on the team has preprocessed the text data.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Karthik** | **Jun Ming** | **Pin Shien** |
| **Data Collection** | * [trustpilot.com](https://www.trustpilot.com/) * selenium | * [airlinequality.com](https://www.airlinequality.com/) * selenium and beautiful soup | * [airlineratings.com](https://www.airlineratings.com/airline-passenger-reviews/) * beautiful soup |
| **Data Cleaning & Preprocessing** | * Remove null values and duplicates * Removing non-English sentences * Lower casing * Remove URL, HTML tags, extra spaces, money and username * Correcting any spelling mistakes * Expanding short form slang and contractions * Negation Handling * Normalize exaggerated text * Convert emojis and emoticons to text * Remove numbers and stopwords, frequent words, and rare words * Lemmatization + POS Tagging | * Removing null values & duplicates * Outliers in text data * Lower casing * Removal of stopwords and punctuations * Removal of numbers and mixed words (numbers and letters) * Tokenize again using n-grams to define the compound words * Lemmatization * Coming out with visualizations | * Removing duplicates * Removing Null values * Removing stopwords * Removing punctuations * Lowercasing * Tokenization * Lemmatization * Stemming |
| **Feature Engineering** | * Feature Engineering (word count, sentence length, unique words, etc) * Label encoding target column (NPS\_category) * Count Vectorizer, TFIDF, Delta TFIDF * Class weights, Smote, Adasyn for imbalance | * Label encoding target column (NPS\_category) * Count Vectorizer & TFIDF * Class weights & Oversampling | * Label encoding target column (NPS\_category) * Count Vectorizer & TFIDF * Smote |

## Model Development

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Karthik** | **Jun Ming** | **Pin Shien** |
| **Models Used** | * Logistic Regression * Decision Tree * Random Forest * Support Vector Machine (SGD w/ hinge loss) * XGBoost * Catboost * Adaboost * LightGBM | * Logistic Regression * Decision Tree * Random Forest * Support Vector Machine * Naïve Bayes | * Logistic Regression * Decision Tree * Random Forest * Support Vector Machine * Naïve Bayes |
| **Rationale** | * Couldn’t use naïve bayes since delta tfidf returns the negative values in output. One workaround to this is to use Gaussian NB but it doesn’t support class weights. * Stochastic Gradient Descent with hinge loss was used for faster training time using SVM (handles data in batches). * Multiple boosting algorithms were experimented with to see effectiveness in solving the problem. |  | * Basic models that we learnt in NLP. * Appropriate for evaluation. |
| **Training Process** | * Experimented with: * TFIDF with weighted classes * TFIDF with additional meta features, unigrams and bigrams * Count vectorizer with weighted classes * Count vectorizer with additional meta features, unigrams and bigrams * Delta TFIDF with class weights, SMOTE, ADASYN. | * Experimented with: * TFIDF with weighted classes * Count vectorizer with weighted classes * TFIDF oversampling * Count vectorizer with oversampling | * Count vectorizer with SMOTE * TFIDF with SMOTE |

Looking at the 4 key evaluation metrics in detail:

* **Accuracy:** Measures the proportion of correctly classified instances. In this context, it represents the overall correctness of your model in predicting whether a review is a Promoter or a Non-Promoter (Detractor or Neutral). A higher accuracy indicates that your model is making correct predictions more often. Since the data is imbalanced, accuracy is not the best metric to make use of.
* **Precision:** Measures the proportion of true positive predictions among all instances predicted as belonging to a specific class. In this case it would indicate the percentage of correctly identified Promoters out of all instances that your model classified as Promoters.
* **Recall:** Measures the proportion of true positive predictions among all actual instances. In this context, it would indicate the percentage of correctly identified Promoters out of all actual Promoter reviews.
* **F1-Score:** Measures the harmonic mean of precision and recall and provides a balance between the two metrics. It is particularly useful when you want to find an optimal balance between minimizing false positives (achieving high precision) and minimizing false negatives (achieving high recall).

In our case, it is important to achieve a **high recall**, especially for the detractor class, as it means the model is accurately able to predict which class each review belongs to. Since the dataset is imbalanced the **macro-average recall** will be looked at as it considers all the classes equally. This will then allow them to take more decisive action on reducing the detractors, converting the passives, and maintaining the promoters.

# **Results**

## Consolidated Test Data outcomes

Karthik’s Best Model Performance on Test Data

***Model:*** *SVM (SGD with hinge loss),* ***Vectorization Technique:*** *Delta-TFIDF*

***Best Parameters:*** *{'alpha': 7.45934328572655e-06, 'eta0': 0.05669849511478854, 'l1\_ratio': 0.7319939418114051, 'learning\_rate': 'optimal', 'penalty': 'l2', 'power\_t': 0.24041677639819287}*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** |
| **Promoters (2)** | 100% | 100% | 100% |
| **Passives (1)** | 96% | 100% | 98% |
| **Detractors (0)** | 100% | 84% | 91% |

Jun Ming’s Best Model Performance on Test Data

***Model:*** *Random Forest,* ***Vectorization Technique:*** *TFIDF*

***Best Parameters:***

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** |
| **Promoters (2)** | 92% | 99% | 95% |
| **Passives (1)** | 83% | 7% | 12% |
| **Detractors (0)** | 87% | 63% | 73% |

Pin Shien’s Best Model Performance on Test Data

***Model:*** *Logistic Regression (trained on SMOTE),* ***Vectorization Technique:***

***Best Parameters:*** *{}*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** |
| **Promoters (2)** | 74% | 78% | 76% |
| **Passives (1)** | 10% | 37% | 15% |
| **Detractors (0)** | 98% | 88% | 93% |

## Model Outcome Comparison

In this section, we analyze and compare the performance of various text processing and classification techniques applied to a dataset with three target classes: Promoters (class 0), Passives (class 1), and Detractors (class 2). We focus on the impact of different feature engineering strategies on the model's ability to predict these classes.

### **Karthik**

Bag of Words (BoW) with Logistic Regression

The initial approach utilized a Bag of Words model trained on logistic regression with class weights. This was specifically designed to address the imbalance in the dataset. Despite these measures, the model's performance on classes other than Promoters was suboptimal, particularly for Passives. The macro average recall stood at 64%, only slightly surpassing the baseline of 50%.

**BoW Classification Report (Logistic Regression with Class Weights):**

* Class 0 (Promoters): Precision at 0.97, recall at 0.90, and F1-score at 0.93.
* Class 1 (Passives): Precision at 0.10, recall at 0.22, and F1-score at 0.14.
* Class 2 (Detractors): Precision at 0.66, recall at 0.80, and F1-score at 0.72.
* Overall accuracy stood at 87%.

BoW with Text Meta Features

Integrating text meta features into the BoW representation resulted in a marginal improvement for Promoters and Detractors. However, the minor performance increase did not substantiate the complexity added by the increased dimensionality.

**BoW with Text Meta Features Classification Report:**

* Modest improvements in precision and recall were noted for Promoters and Detractors.
* Overall accuracy saw a slight increase to 88%.

BoW with Unigrams and Bigrams

Expanding the feature set to include unigrams and bigrams, fed into the CountVectorizer with an ngram range of (1, 2), led to improvements in recall and F1-score for Promoters, precision for Passives (albeit with reduced recall), and uplifts in all performance metrics for Detractors. This suggests that capturing context through bigrams is crucial for predicting Detractors.

**BoW Classification Report with Unigrams and Bigrams:**

* Class 0 (Promoters): Enhanced performance with precision at 0.96 and F1-score at 0.96.
* Class 1 (Passives): Precision improved to 0.18, but recall dropped to 0.12.
* Class 2 (Detractors): Gains across all metrics with precision at 0.76 and F1-score at 0.78.
* Overall accuracy rose to 92%.

TF-IDF Vectorization

Implementing TF-IDF vectorization, the performance for Passives showed improvement in recall but at the cost of lower precision across models. This method did not significantly alter the outcome for Promoters but yielded better recall for Detractors.

**TF-IDF Classification Report:**

* Class 1 (Passives): Higher recall at 0.29 but precision remained low at 0.10.
* Class 2 (Detractors): Consistent improvement with recall at 0.80.
* Overall accuracy was at 86% for pure TF-IDF and 87% with meta text features.

TF-IDF with Unigrams and Bigrams

Further enhancement using TF-IDF with unigrams and bigrams showed a mixed impact: while it improved recall and F1-scores for Promoters, it did not improve the performance for Passives.

**TF-IDF Classification Report with Unigrams and Bigrams:**

* Class 0 (Promoters): Maintained high performance with precision at 0.96 and F1-score at 0.96.
* Class 1 (Passives): Precision increased slightly to 0.20, but recall decreased to 0.10.
* Class 2 (Detractors): Showed consistent improvement across metrics.
* Overall accuracy achieved 92%.

tfidf with meta text features:

precision recall f1-score support

0 0.97 0.89 0.93 11894

1 0.10 0.28 0.15 338

2 0.68 0.80 0.73 1920

accuracy 0.87 14152

macro avg 0.58 0.66 0.60 14152

weighted avg 0.91 0.87 0.88 14152

tfidf with unigrams and bigrams:

TF-IDF Classification Report:

precision recall f1-score support

0 0.96 0.96 0.96 11894

1 0.20 0.10 0.14 338

2 0.75 0.81 0.78 1920

accuracy 0.92 14152

macro avg 0.63 0.62 0.62 14152

weighted avg 0.91 0.92 0.91 14152

### **Jun Ming**

A screenshot of a computer

Description automatically generated

*Figure 1 – Random Forest not tuned*

Using precision as the main evaluation, we pick the random forest using tfidf and weighted class as it has the overall highest precision out of all of the models. Precision for class 0: 0.92, Precision for class 1: 0.83, Precision for class 2: 0.87. This is because the f1-score and recall of all the models built was roughly the same. This is also because we want to ensure that when we predict a class it is indeed that class as that is what the precision looks at and tells us. Thus, putting the precision score as the main evaluation is justified. According to this using random forest along with tfidf and weighted class gives us the best model.

Choosing between tuned and untuned models:

|  |  |
| --- | --- |
| A graph of a function  Description automatically generated |  |
| *Figure 2 - Tuned random forest with tfidf and weighted class* | *Figure 3 – Untuned random forest with tfidf and weighted class* |

The Receiver Operating Characteristic (ROC) curve is a graphical plot that illustrates the diagnostic ability of a binary classification model across various threshold settings. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at different threshold values.

* True Positive Rate (TPR), also known as sensitivity or recall, measures the proportion of positive instances that are correctly identified by the model.
* False Positive Rate (FPR) measures the proportion of negative instances that are incorrectly classified as positive by the model.

The Area Under the ROC Curve (AUC) quantifies the overall performance of the binary classification model. It represents the probability that the model will rank a randomly chosen positive instance higher than a randomly chosen negative instance. AUC values range from 0 to 1, where higher values indicate better performance. An AUC of 0.5 suggests a model with no discriminatory power (random guessing), while an AUC of 1 represents a perfect model that perfectly separates the classes.

For ROC, the closer it is to creating a right angle at the top left corner the better as the tradeoff between True positive rate and False Positive rate is none which means that it can have a 100% True positive rate but 0% False Positive rate. which we cannot really tell the difference thus we will look at the AUC.

AUC:

* Class 0 has a very high AUC of 0.93, indicating excellent performance in distinguishing between class 0 and other classes.
* Class 1 has a lower AUC of 0.78, suggesting less discriminatory power compared to class 0 and class 2.
* Class 2 has a high AUC of 0.94, indicating strong performance in distinguishing between class 2 and other classes.

From the ROC curve, we can tell that there is a tradeoff between Class 0 (Promoter) and Class 1 (Neutral) when tuning. Although class 1 increases, Class 0 (Detractor) AUC decreases. This is not beneficial to us as our main focus is to find the promoters and detractors of the reviews, the neutral class (Class 1) is important but not as important as Class 0 (Detractor) or Class 2 (Promoter). Thus we wouldn't want to sacrifice the AUC of Class 0 (Detractor) and Class 2 (Promoter) for Class 1 Neutral). Thus, using this set of metrics, the base model would perform better in our business case compared to the tuned model.

A screenshot of a graph

Description automatically generated

*Figure 4 – Model on Validation Data*

These metrics provide insights into the performance of the model for each class. Class 0 (Detractor) has high precision, recall, and F1-score, indicating strong performance. Class 1 (Neutral), however, shows low recall and F1-score, suggesting that the model struggles to correctly identify instances of this class. Class 2 (Promoter) exhibits moderate performance with reasonably high precision, recall, and F1-score.

|  |  |
| --- | --- |
|  |  |
| *Figure 5 – Confusion Matrix* | *Figure 6 – Misclassification Scores* |

From the confusion matrix above, the overall misclassification rate is very low. However, when we examine the misclassification rate for each class, it seems that Class 0 and 2 have relatively low rates compared to Class 1. This suggests that the model struggles to predict Class 1 as effectively as the other two classes. However, this isn't a significant issue because:

1. Neutral classes (Class 1) do not appear frequently.
2. They have less impact than the other two classes.

Thus, this model proves to be quite effective for our business scenario. In our business context, we are particularly interested in predicting detractors. Detractors are defined as individuals who actively undermine the business. Predicting these detractors and addressing their criticisms can help the business improve its overall quality. In this case, the misclassification rate for detractors is extremely low, at around 3%, which is excellent for the business. Additionally, the high recall, precision, and F1 score from above further support the notion that the model excels at detecting detractors

### **Pin Shien**

|  |  |
| --- | --- |
|  | A screenshot of a computer screen  Description automatically generated |
| *Figure 7 – Logistic Regression (best model)* | *Figure 8 – SMOTE + tuning model* |

**Strengths:**

* High Precision for Detractors (0.98): The model shows very high precision for classifying detractors, indicating that when it predicts a detractor, it is usually correct.
* High Recall for Detractors (0.88): The model also has a relatively high recall for detractors, indicating that it effectively captures a large portion of the actual detractors.
* High F1-Score for Detractors (0.93): The high F1-score for detractors suggests a good balance between precision and recall for this class.

**Weaknesses:**

* Low Precision for Neutral (0.10): The model's precision for neutral instances is very low, suggesting that when it predicts a neutral instance, it is often incorrect.
* Low Recall for Neutral (0.37): The recall for neutral instances is also relatively low, indicating that the model fails to capture a significant portion of the actual neutral instances.
* Low F1-Score for Neutral (0.15): The low F1-score for neutral instances highlights the poor performance in both precision and recall for this class.

# **Discussion**

# **Conclusion**