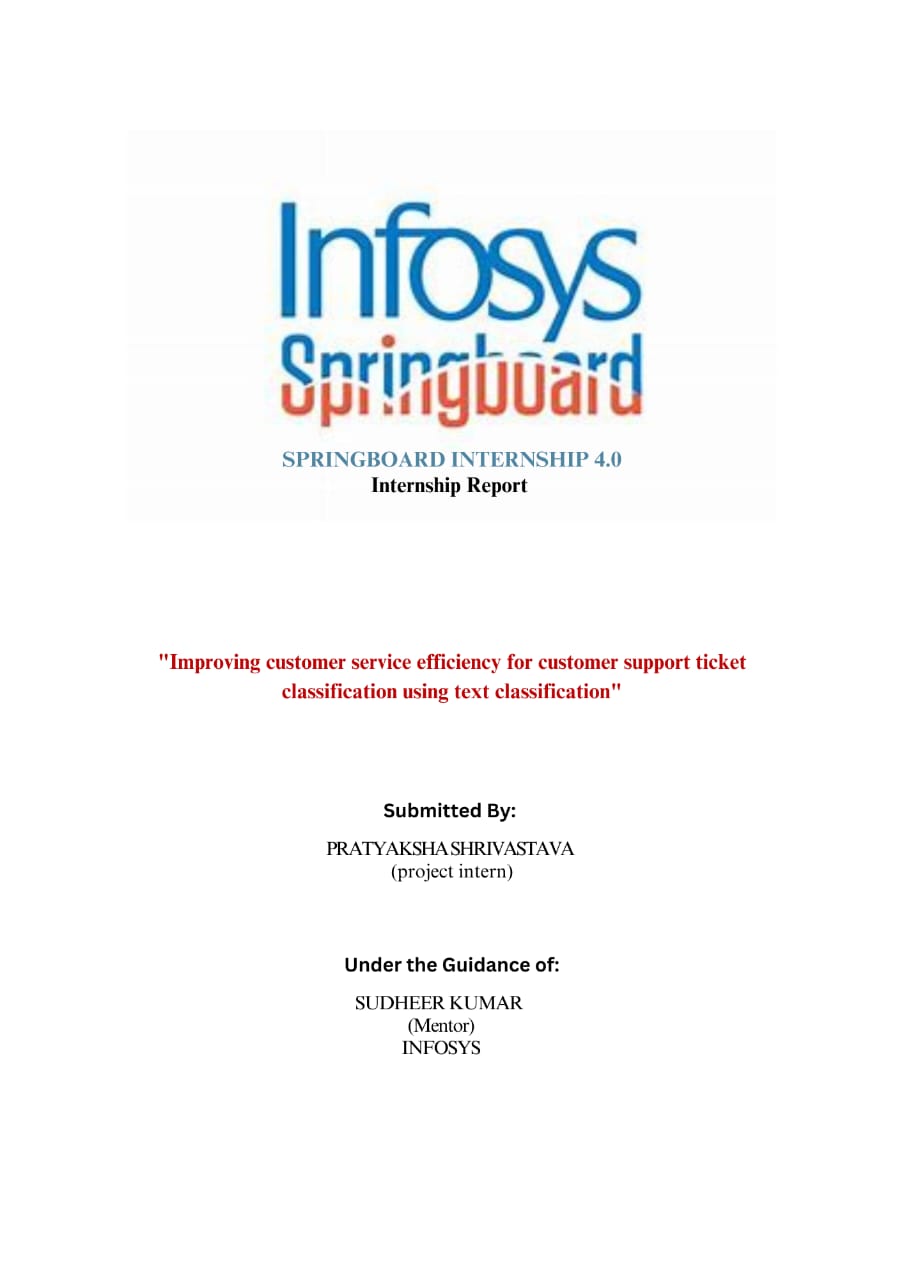
******

***Acknowledgement:***

I would like to convey our heartfelt gratitude to Sudheer Kumar Y'sir for their immense support, other important help, and many technical suggestions during the entire making of the project. I am very grateful to Infosys Springboard Program 4.0 for their unconditional support during the course of this time. I would also like to give my deep thanks to all the members of my batch of Infosys Springboard Program 4.0 for their direct or indirect contribution to my project work and for providing us with a friendly environment in the meeting. Finally, I would also like to give my deepest thanks to all my family members and all my friends for their constant moral support.***Abstract:***

Automated text classification has been considered as a vital method to manage and process a vast amount of documents in digital forms that are widespread and continuously increasing. In general, text classification plays an important role in information extraction and summarization, text retrieval, and question answering. This paper illustrates the text classification process using machine learning approaches. The references cited cover the major theoretical issues and guide the researcher to interesting research directions

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **Topic** | **Page No.** |
| **1.** | **Introduction** | **1** |
| **2.** | **Business Use Case** | **2-3** |
| **3.** | **Sentiment Analysis** | **4-7** |
| **4.** | **Implementation Steps Of Text Classification** | **8** |
| **5.** | **About Dataset** | **9** |
| **6.** | **Data (Text) Pre-processing** | **10-11** |
| **7.** | **Modeling Approach** | **12-13** |
| **8.** | **Model used** | **14-16** |
| **9.** | **Hyperparameter tuning** | **17-18** |
| **10.** | **Result** |  |
| **11.** | **Recommendation and Thankyou (Q&A)** |  |
| **12.** |  |  |

***Introduction***

***Customer Support Ticket Classification***

Customer support is an essential function in businesses, bridging the gap between customers and companies. Effective customer support not only resolves issues but also enhances customer satisfaction and loyalty. A critical aspect of modern customer support systems is the classification of support tickets. This process involves categorizing incoming support requests to streamline their resolution. Efficient ticket classification can significantly improve response times, optimize resource allocation, and enhance overall service quality. Customer support ticket classification is a crucial component in managing and enhancing the efficiency of customer service operations. Leveraging machine learning and natural language processing for automated classification not only improves response times and accuracy but also provides valuable insights for business improvement. By enabling consistent, accurate, and scalable ticket processing, automated classification enhances customer satisfaction and optimizes resource allocation. As technology evolves, the capabilities of these systems will continue to expand, setting new standards for customer support excellence. This evolution promises a more responsive, efficient, and insightful customer support framework, ultimately fostering stronger customer relationships and driving business success.

***Business Use Case:***

***Improving Customer Service Efficiency***

***Problem Description:*** A company receives a large volume of customer support tickets through various channels (email, chat, web forms). These tickets cover a wide range of topics, such as technical issues, billing inquiries, product feedback, and general inquiries. Manually categorizing and prioritizing these tickets is time-consuming and error-prone. The company wants to automate this process by classifying incoming support tickets into relevant categories.

***Solution***: Implement a text classification system that can automatically assign predefined tags or labels to each customer support ticket based on its content. The system should be able to categorize tickets into specific topics (e.g., technical support, billing, refunds, and account management) and urgency levels (e.g., high priority, medium priority, low priority).

***Business Use Case Benefits:***

***Efficiency:*** By automating ticket classification, the company can save time and resources. Support agents no longer need to manually read and categorize each ticket, allowing them to focus on resolving customer issues promptly.

***Improved Response Times:*** High-priority tickets can be addressed more quickly, ensuring timely responses to urgent customer inquiries.

***Resource Allocation:*** The system helps allocate support resources effectively. For instance, technical support specialists can handle technical issues, while billing specialists can handle billing-related inquiries.

***Data-Driven Insights:*** Analysing categorized tickets provides valuable insights into common customer pain points, frequently asked questions, and areas for process improvement.

***Consistent Service:*** Customers receive consistent and accurate responses, regardless of the support agent handling their ticket.

***Sentiment Analysis:***

Sentiment analysis is becoming very important to study growing opinions faster and faster within social media and other sites, the huge explosion in information in recent years in the sites of communication, air traffic and alternative markets, all this huge amount of information cannot be controlled and analysed used the traditional way, so the scientists and researchers developed a high-efficiency techniques to deal with this data. This requires the SA to process data and know its polarity to determine the right decision. SA involves five steps to process data; those are data collection, text preparation, sentiment detection, sentiment classification, and presentation of output.

***1. Data Collection:***

The collection of data from sources like user groups, Twitter, Facebook, blogs and commercial website such as amazon.com and alibaba.com, etc. This data cannot be analysed using traditional methods like scanning, text analysis, and language processing, which is used for extraction and classification.

***2. Data pre processing:***

Before analysing the data, it’s essential to clean and pre-process it. Remove any noise, irrelevant information, or formatting issues.

Common pre-processing steps include converting text to lowercase, removing special characters, handling misspellings, and eliminating stop words (common words like “the,” “and,” etc.).

***3. Analyse Your Data:***

Apply sentiment analysis techniques to the cleaned data. There are two main approaches:

**Machine Learning-Based Approach:**

Train a machine learning model using labeled data (supervised learning).

The model learns to classify text into positive, negative, or neutral sentiments based on features extracted from the text.

**Lexicon-Based Approach:**

Use predefined lexicons or dictionaries containing words and their associated sentiment scores.

Assign sentiment scores to individual words in the text and aggregate them to determine the overall sentiment.

***4. Data visualization:***

Create visualizations to understand the sentiment distribution.

Bar charts, word clouds, or sentiment heat maps can help you visualize positive, negative, and neutral sentiments.

***Step by step process of sentimental analysis:***

***1. Text Pre-processing:***

The first step is to clean and prepare the text data. This ensures that the text is in a consistent format for analysis.

In your code, the preprocess\_text function performs the following tasks:

Converts the text to lowercase.

Removes URLs (links).

Replaces newline characters with spaces.

Removes words containing numbers.

Removes extra spaces.

Removes special characters.

Removes common stop words (e.g., “the,” “and,” etc.).

Applies stemming (reducing words to their root forms) and lemmatization (reducing words to their base forms).

***2. Sentiment Lexicon:***

A sentiment lexicon (dictionary) is used to associate words with sentiment scores. In your code, you’ve defined a lexicon containing words like “good,” “bad,” and “neutral,” along with their corresponding sentiment scores (e.g., 1 for positive, -1 for negative, and 0 for neutral).You can expand this lexicon by adding more words and scores as needed.

***3. Tokenization:***

Tokenization involves splitting the cleaned reviews into individual words (tokens).The word\_tokenize function from the NLTK library is used for this purpose.

***4. Sentiment Score Calculation:***

For each review, the sentiment score is calculated based on the lexicon. The analyze\_sentiment function iterates through the tokens in a review and accumulates the sentiment scores for words found in the lexicon. The resulting score represents the overall sentiment of the review.

***5. Sentiment Labels:***

The sentiment scores are mapped to sentiment labels:

Positive if the score is greater than 0.

Negative if the score is less than 0.

Neutral if the score is 0.

The labels are stored in a new column called 'review\_sentiment'.

***6. Visualization:***

The code creates a bar chart to visualize the distribution of sentiments in the reviews.

Positive sentiments are shown in green, negative in red, and neutral in grey.

***Implementation Steps Of Text Classification:***

***1. Data Collection:*** Gather historical support tickets with their corresponding labels (categories and urgency levels).

***2. Pre-processing:*** Clean and pre-process the text data (Lower Case, Remove links, Remove next lines, Remove words containing numbers, Remove extra spaces, Remove special characters, Remove stop words, Stemming, Lemmatization).

***3. Feature Extraction:*** Convert text into numerical features (**TfidfVectorizer**).

***4. Model Selection:*** Choose an appropriate machine learning algorithm (Multinomial Naive Bayes, LogisticRegression, RandomForestClassifier, support vector classifier, KNeighborsClassifier).

***5. Training***: Train the model using labelled data.

***6. Evaluation***: Evaluate the model’s performance using metrics such as accuracy, precision, recall, and F1-score.

***7. Deployment***: Deploy the trained model in the production environment to automatically classify incoming support tickets.

***About Dataset:***

Dataset consist of following column:

1. Label: Integer

2. Text: String

The total length of the dataset is 16001 data. It consist of labels such as 0:sadness , 1:joy , 2:love , 3:anger , 4:fear , 5:surprise.The count of sadness is: 4666,joy is: 5362, love is: 1304,anger is: 2159,fear is: 1937,surprise is : 572.

A few classifications, similar to surprise and love, have less information compared with others, similar to satisfaction and misery. This imbalance should be tended to during model preparation. This is vital to guaranteeing that the classifier functions admirably for all feelings. To deal with this irregularity, strategies include, for example, adding more information or changing the quantity of tests. At first, we dealt with the nostalgic investigation utilizing different datasets, which gave us a better understanding of the text grouping.

***Data (Text) Pre-processing:***

***Lower Case:*** Convert all text to lowercase to ensure consistency and avoid duplication of words that differ only in case.

***Remove Links:*** Eliminate any hyperlinks present in the text as they typically do not contribute to the meaning of the text and can be distracting.

***Remove New Lines (\n):*** Strip out newline characters to ensure that text is continuous and easier to process.

***Words Containing Numbers:*** Exclude words that contain numbers since they often represent identifiers or measurements rather than meaningful words.

***Extra Spaces:*** Trim excess spaces to standardize the text and ensure consistent tokenization.

***Special Characters:*** Remove special characters like punctuation, symbols, or emoji’s, as they may not contribute to the semantics of the text.

***Removal of Stop Words:*** Filter out common words (stop words) like "the," "is," "and," which occur frequently but often do not carry significant meaning.

***Stemming:*** Reduce words to their base or root form, removing suffixes and prefixes. For example, "running" becomes "run." This helps in reducing the dimensionality of the feature space and capturing the essence of the word. Popular stemming algorithms include Porter Stemmer, Snowball Stemmer, and Lancaster Stemmer.

***Lemmatization:*** Similar to stemming, but instead of just removing prefixes and suffixes, lemmatization maps words to their base or dictionary form (lemma). For example, "better" becomes "good." This approach ensures that the transformed words are valid lemmas, which can improve interpretability.

***Modeling Approach:***

**1. Data (Text) Pre-processing:**

Lower Case

Remove links

Remove next lines (\n)

Words containing numbers

Extra spaces

Special characters

Removal of stop words

Stemming

Lemmatization

***2. Featuring Engineering:***

• The TfidfVectorizer is used to convert the pre-processed text into numerical features using the Term Frequency-Inverse Document Frequency (TF-IDF) representation.

• X contains the TF-IDF features, and y contains the corresponding labels (target variable).

**3. Model Building:**

1. Divide the dataset in to Train (70%), Test (20%) and Validation (10%) datasets.

2. Build at least 3 classification models

**Step 1:** Build model 1 and generate the classification report (Performance metrics using Confusion Metrics) for both

Training and Test datasets.

**Step 2:**

• Use grid search or binary search for Hyperparameter Tuning.

• Use at least 2 values for each hyperparameters.

• Choose the best model parameters based on grid search and generate the classification report (Performance metrics using Confusion Metrics) for both Training and Test datasets.

Step 3: Repeat step 1 and 2 for Model 2 and Model 3 as well.

Step 4: Now choose the final model based on the classification report (Performance metrics using Confusion Metrics) for both Training, Test and validation datasets.

**4. Data Visualization: Input and Output plots**

***Model used:***

***1. Multinomial Naive Bayes (MNB):***

Multinomial Naive Bayes is a probabilistic classifier based on Bayes’ theorem.

It’s particularly useful for text classification problems in NLP.

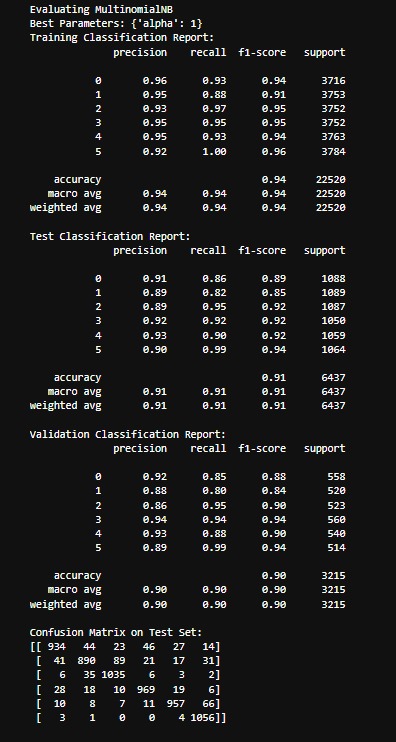
MNB assumes that features (e.g., word counts) are conditionally independent given the class label.

It’s well-suited for data with features representing discrete frequencies or counts (e.g., word occurrences).

**Hyperparameters**:

Alpha is Regularization strength (similar to C in Logistic Regression).

Suggested values: [0.01, 0.1, 1]



***2. Logistic Regression:***

Logistic Regression is a popular supervised learning algorithm used for binary classification (where the target variable has two classes) and multiclass classification (where the target variable has more than two classes).

Despite its name, it is used for classification, not regression.

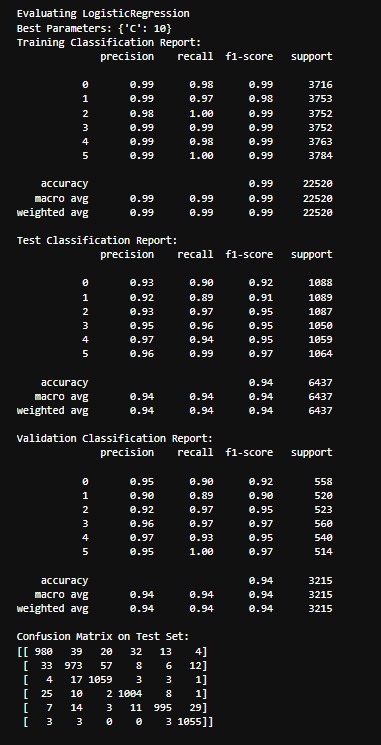
The goal of logistic regression is to predict the probability that an instance belongs to a particular class.

It models the relationship between a set of input features (predictors) and a binary output (0 or 1) using the logistic function (also known as the sigmoid function).

**Hyperparameters**

C: Regularization parameter controlling the inverse of regularization strength.

Suggested values: [0.1, 1, 10]



***3. RandomForestClassifier:***

The RandomForestClassifier is part of the ensemble learning family in skit-learn.

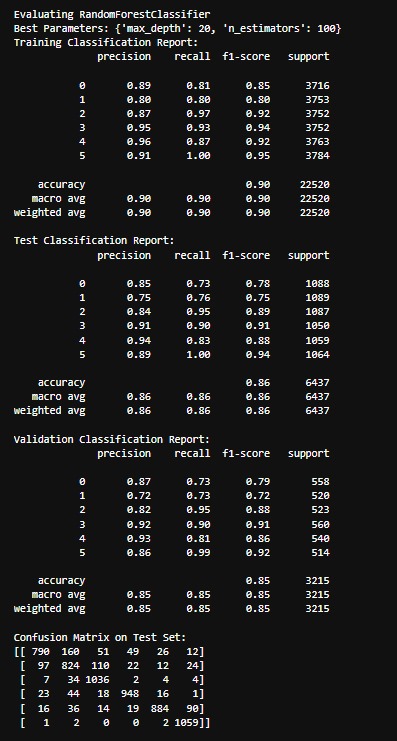
It is a meta-estimator that combines multiple decision tree classifiers to improve predictive accuracy and control overfitting.

***Hyperparameters***

Number of decision trees (controlled indirectly by the number of estimators).

max\_depth: Maximum depth of each tree.

Suggested values: n\_estimators: [50, 100], max\_depth: [10, 20].



***4. Support vector classifier:***

The Support Vector Classifier (SVC), also known as Support Vector Machine (SVM), is a powerful supervised learning algorithm used for classification tasks.

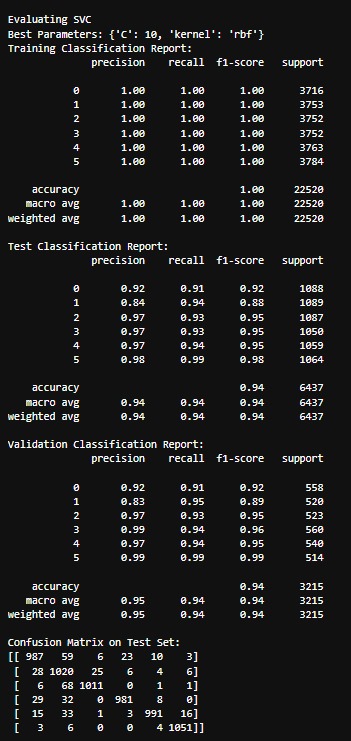
It aims to find a hyperplane (a decision boundary) in an N-dimensional space (where N represents the number of features) that distinctly separates data points belonging to different classes.

***Hyperparameters:***

C: Regularization parameter.

Kernel: Kernel function (linear or radial basis function).

Suggested values: C: [0.1, 1, 10] , kernel: ['linear', 'rbf'].



***5. KNeighborsClassifier:***

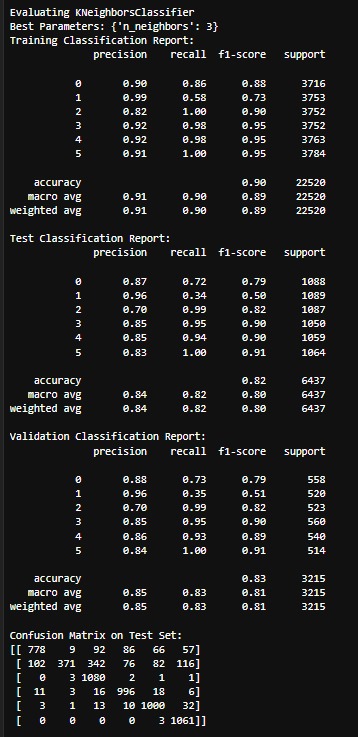
The K-Nearest Neighbors (KNN) algorithm is a non-parametric and instance-based supervised learning method.

It makes predictions based on the similarity of an input sample to its k nearest Neighbors in the training data.

***Hyperparameters:***

n\_neighbors: Number of neighbours to consider.

Suggested values: [3, 5, 7]



***Hyperparameter tuning:***

Hyperparameter tuning refers to the process of selecting the optimal values for a machine learning model’s hyperparameters.

The goal of hyperparameter tuning is to find the best combination of hyperparameter values that maximizes the model’s performance.

***Hyperparameters:***

Hyperparameters are settings or parameters that control the learning process of a machine learning model.

Unlike model parameters (such as weights learned during training), hyperparameters are not learned from the data but are set before training begins.

Examples of hyperparameters include:

Learning rate

Number of hidden layers in a neural network

Regularization strength

Kernel size in a support vector machine (SVM)

Number of neighbors in k-nearest neighbors (KNN)

**Hyperparameter Tuning python**

def build\_and\_evaluate\_model(model, param\_grid):

...

This function handles the training and evaluation of a model with hyperparameter tuning:

**Grid Search for Hyperparameter Tuning python:**

grid\_search = GridSearchCV(model, param\_grid, cv=5, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

GridSearchCV is used to perform an exhaustive search over specified parameter values (defined in param\_grid) for the given model. It uses 5-fold cross-validation to evaluate each combination of parameters.

***Conclusion:***

### Conclusion

After evaluating multiple classification algorithms, including Logistic Regression, Support Vector Machine (SVM), Naive Bayes, Random Forest, and K-Nearest Neighbors (KNN), Logistic Regression emerged as the best-performing model for our dataset. Logistic Regression demonstrated superior accuracy and robustness, effectively capturing the underlying relationships between features and the target variable. Despite the presence of several strong contenders, Logistic Regression outperformed others, particularly in terms of its balance between bias and variance, and its ability to generalize well to unseen data. Additionally, the interpretability of Logistic Regression makes it a valuable tool, providing clear insights into the influence of individual features on the outcome. These qualities make Logistic Regression an ideal choice for this classification task, ensuring both high performance and practical applicability.

