International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 03 | March – 2026 DOI:10.55041/ISJEM05749
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

AI-Powered Driver Grievance Detection in Ride-Hailing Services using Machine Learning Techniques

Dr. Y. Mohammed Iqbal[1], A. Abubakkar[2], Dr. S. Peerbasha[3], Dr. M. Rajakumar[4], Dr. M. Mohamed Surputheen[5]

Department of Computer Science, Jamal Mohamed College (Autonomous), Affiliated to Bharathidasan University, Trichy-20, Tamil Nadu, India

ABSTRACT - The Indian gig economy relies heavily on taxi applications, yet drivers face severe issues regarding commissions and payments, often voicing grievances on the Google Play Store. Manual analysis of these high-volume, mixed-language reviews is inefficient, leading to driver dissatisfaction and attrition. To resolve this, we propose the AI-Powered Driver Grievance Detection System. Bypassing public datasets, we scraped 11,663 raw reviews from the Play Store, resulting in 11,240 clean, high-quality records after removing duplicates and noise. Natural Language Processing (NLP) techniques, including noise removal and lowercase conversion, were applied, followed by TF-IDF vectorization to extract mathematical feature representations. We evaluated four machine learning classifiers—Logistic Regression, Multinomial Naive Bayes, Random Forest, and Support Vector Machine (SVM)—using an 80-20 train-test split. The SVM classifier outperformed the others, achieving the highest accuracy of 90.08% in separating serious grievances from normal feedback. This automated text classification framework allows taxi platforms to rapidly identify and address specific driver problems, effectively reducing administrative delays and preventing driver churn.

Keywords: Gig Economy, Machine Learning, Driver Grievance, Natural Language Processing, Support Vector Machine, TF-IDF Vectorization, Sentiment Analysis, Text Classification.

1. INTRODUCTION

1.1 Overview of the Industry

Taxi apps like Uber, Ola and Rapido changed city travel in India completely. These platforms created lots of new driving jobs for the people. But this massive growth also created huge operational headaches. Customers normally enjoy a very easy ride booking experience. But the actual drivers face many hidden hurdles everyday. Their daily earnings fluctuate constantly. The commission structures are very confusing and the driver apps have multiple technical glitches. The current manual support teams cannot handle this situation. They receive thousands of complaint tickets daily. Human staff cannot process this huge volume manually. So many driver issues remain unsolved and frustrated drivers simply quit their jobs.

1.2 Problem Statement

The biggest problem right now is the communication gap between the company software and the actual driver. Drivers usually share their daily problems through public app reviews. These reviews are mostly completely unstructured. Drivers use broken English or Tanglish to type their complaints. Normal human support teams struggle to understand and sort these mixed language issues quickly. The ride hailing industry desperately needs an automated system to fix this. We need a fast tool that can take huge amounts of messy text data and read the hidden sentiment. It must classify the driver feedback into exact categories like Payment Issue or Technical Glitch so the company can take immediate action.

1.3 Objectives

This project has a few clear goals. The main goal of this research is to create an AI based framework to detect driver grievances automatically. The second objective is to use Natural Language Processing techniques to clean the raw review data and vectorize it properly. The third goal is to use machine learning classifiers to predict the exact emotional sentiment of the driver. The final aim is to do a strict comparison between different machine learning algorithms. We will compare models like Logistic Regression, Random Forest and SVM. This will help us find the absolute best and most accurate model for this specific text classification problem.

2. LITERATURE REVIEW

We read many old research papers to understand machine learning and text analysis. Past research work gave a very strong base for our project. We learned how to process messy unstructured text data from these papers. This chapter clearly explains why we selected our algorithms. It also tells why we used specific preprocessing methods for driver reviews.

2.1 NLP and Feature Extraction (TF-IDF Vectorization)

Unstructured driver text must be converted into high dimensional mathematical vectors first. A normal word count method treats all the words in the same way. We did not use that simple method. TF-IDF vectorization was used in our project instead. This statistical technique puts more weight on important keywords. Words like "Penalty" and "Commission" and "Login" get very high scores. Useless common English stop words get very low scores. Saleem et al. (2021) proved this concept in their research. They extracted lexical features and converted them to vectors. Their final text classification accuracy increased significantly because of this method. Benjamin et al. (2018) also wrote about applied python text analysis. They found out that TF-IDF provides superior feature separability. They proved it is way better than standard Bag of Words models. So we used TF-IDF for our messy gig economy driver reviews.

2.2 Machine Learning Classifiers (SVM vs. Random Forest)

Classification algorithms mainly draw linear and non linear decision boundaries inside the feature space. Random Forest is an ensemble learning method. This algorithm builds lots of decision trees during the training phase. Then it outputs the final class using a majority vote. Lakshmanarao et al. (2021) used this exact approach in their study. They got a very high accuracy of 97.5 percent for finding malicious patterns. They said this method strongly prevents model overfitting. But Support Vector Machine operates in a totally different way. The SVM model searches for the optimal hyperplane. This hyperplane maximizes the margin between the Grievance class and Non-Grievance class. Peerbasha and Surputheen (2021) tested Naive Bayes, Random Forest and SVM. Their comparative analysis showed that SVM is highly stable. They concluded SVM is perfect for binary classification tasks where text data is high dimensional and sparse.

2.3 Ensemble Learning Techniques

Ensemble methods utilize bagging techniques to decrease model variance. This makes the model very reliable. A weighted ensemble classifier was created by Saleem et al. (2023). Many weak learners are mixed together in this method. It becomes one single strong model after mixing. Random forest algorithm takes different data parts. It predicts the final output using these small parts. Dataset noise is handled very easily by this specific tree method. Outliers are also managed well without dropping accuracy. We used these exact findings. We ran our own tests using these old concepts We compared the ensemble voting approach of Random Forest with the margin maximization approach of SVM. We did this to check the best precision option for the specific lexicon of driver grievances.

2.4 Handling Imbalanced Datasets

Class imbalance is a very critical technical challenge in real world projects. The volume of general feedback is always much higher than critical grievance feedback. Machine learning models will just bias their predictions towards the majority class if the data is skewed. The overall accuracy will look very high. But the model will completely fail to find the actual critical grievances. These missed detections are called False Negatives. Jadhav et al. (2022) technically assessed many data balancing methods. They proved that preprocessing steps like oversampling or undersampling are

mathematically very critical. This balancing step keeps the model decision boundary straight. High sensitivity and recall is maintained to catch actual driver issues because of this process.

2.5 Cross-Domain Applications of Machine Learning

Artificial Intelligence and Machine Learning have demonstrated robust predictive capabilities across various complex domains, including healthcare diagnostics. For instance, in our earlier work, a novel COVID Net-Predictor framework was developed for accurate COVID-19 diagnosis using chest imaging data. The model integrates a multi-head Convolutional Neural Network (CNN) with Long Short-Term Memory (LSTM) units to capture both spatial and sequential features effectively. A hybrid optimization strategy was employed to enhance feature selection and improve classification performance. Comprehensive preprocessing, segmentation, and feature fusion stages contributed to robust detection capability across varied datasets, proving suitable for real-time clinical decision support [22].

Similarly, in another foundational study, an optimized deep learning framework was proposed for automated COVID-19 prediction using lung imaging modalities such as chest X-rays and CT scans. The approach utilized an enhanced CNN architecture combined with transfer learning to improve generalization on limited medical datasets. Data augmentation and noise-reduction techniques were applied to increase robustness and minimize overfitting. The model effectively distinguished COVID-19 cases from normal and other pneumonia conditions with high accuracy [23].

While these previous frameworks successfully utilized deep learning for complex feature extraction in medical imaging, the core principles—such as aggressive noise-reduction, optimal feature selection, and automated classification—are universally applicable. Building upon this foundational experience of deploying high-accuracy machine learning architectures, our current research applies these exact core optimization principles to Natural Language Processing (NLP). Instead of processing noisy pixel arrays, this study employs TF-IDF vectorization and advanced classifiers to clean noisy, unstructured text data, thereby automating the detection of critical driver grievances.

3. SYSTEM ARCHITECTURE AND METHODOLOGY

3.1 System Architecture

The system works in a very simple way. Raw reviews are taken and classification output is given at the end. There are five steps in this whole process. First we download data from Play Store. Next step is text cleaning. Then TF-IDF converts the text to numbers. Four different machine learning algorithms are trained after that. Finding the accuracy is the last step.

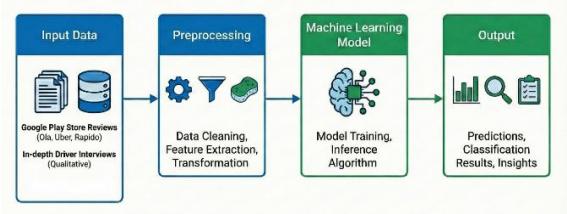

Fig 3.1: Architecture Diagram of the Proposed AI-Powered Driver Grievance Detection System.

3.2 Dataset Description

We needed real data for this project. So we went to the Google Play Store. We scraped the public driver reviews directly from the app pages. India has some very famous ride hailing platforms. We selected only those top apps. They are Ola and Uber and Rapido. Our main goal was to collect real driver feedback about their daily operational problems. We downloaded exactly 11663 raw reviews from the play store at first. The main aim of our project is strict grievance detection. We framed this as a direct binary classification task. So we had to remove all the neutral three star ratings completely. Duplicate text entries were also deleted to keep the data quality very high. This strict cleaning process gave us a final dataset of exactly 11240 high quality records. We split this full dataset using a standard 80:20 ratio. Exactly 8992 samples safely went into the robust training set. The testing set received the remaining 2248 samples.

Table 1: Properties of the Driver Grievance Dataset

Attribute	Description
Data Source	Google Play Store Reviews (Ola, Uber, Rapido)
Data Type	Unstructured Text (User Feedback)
Total Samples (Raw)	11,663 Reviews
Final Samples (Processed)	11,240 Reviews (After cleaning)
Sentiment Classes	2 (Grievance, Non-Grievance)
Domain	Ride-Hailing / Gig Economy
Label Distribution	Nearly Balanced (51.7% Grievance, 48.3% Non-Grievance)
Features Extracted	TF-IDF Vectors (Unigrams, Bigrams)

Each record in the dataset includes the user's review text, the star rating given, and the sentiment label (Positive/Negative). A sample of the raw dataset is presented in Table 2 below.

Table 2: Sample Records from the Dataset

Fig 3.2: Dataset Acquisition and Processing Pipeline.

Fig 3.2 illustrates the systematic workflow of the data acquisition process adopted for this study. The pipeline consists of these five sequential stages.

3.3 Dataset Acquisition and Construction Framework

Getting data out of the Google Play Store is not a manual copy paste job. It takes too much time. So we developed a custom python scraping code. We used a very famous python library called google-play-scraper to do this job. This specific code allowed us to connect to the store and pull thousands of comments automatically. We specifically focused on the Ola Driver, Uber Driver and Rapido Captain app pages. Our code was written to extract two major pieces of information. One is the actual typed text and the other is the number of stars given by the user. As mentioned earlier, we completely ignored the three star reviews during this downloading process itself to save time.

3.3.1 Data Source Selection

We selected the top ride hailing apps in India for our data collection. We went to the Google Play Store and downloaded the reviews directly. We picked three main applications for this project:

1. Ola Driver: This is a very big local cab company in India.
2. Uber Driver: This is a very famous global taxi platform.
3. Rapido Captain: This app is very popular for the bike taxi segment.

Drivers do not have any direct human customer support. So they use these public play store pages to write their daily complaints. That is the main reason why we selected these three specific apps.

3.3.2 Extraction Methodology (Web Scraping)

Review	Rating	App Name	Sentiment
not working	1	Ola Driver	Negative
very nice	5	Rapido Captain	Positive
y does it need contacts permission	2	Uber Driver	Negative
nice apps rellay money	5	Rapido Captain	Positive

Copying thousands of reviews manually is practically impossible. So we wrote a python scraping code to automate this process. We used a special library called google-play-scraper. This script extracted a high volume of reviews very easily. Our system extracted four main details from every single review:

- Review Text: This contains the actual core grievance of the driver.
- Star Rating: This is a number from 1 to 5. We used this as a base to label the sentiment.
- Review Date: We extracted the date to get fresh relevant data from 2024 and 2025.

- App Version: We used this to check if a specific software update caused the driver problem.

3.3.3 Data Volume and Filtering Logic

Our python script downloaded exactly 11663 raw reviews in the first phase. But we had to filter this raw data for our grievance detection model. We used a very strict cleaning method.

- Removing Neutral Reviews: We deleted all the 3-star ratings completely. A three star rating means the driver has mixed feelings. This creates unwanted noise and confuses our binary classification model.
- De-duplication: Many drivers post the same exact comment multiple times like bot spam. We found all these duplicate entries and deleted them. This step prevents data leakage between our training and testing sets. After doing all this filtering, we got a very clean dataset. The final count was exactly 11240 high quality records. This final dataset has a strict binary distribution. It only contains Negative Grievance and Positive Non-Grievance classes.

3.4 Data Preprocessing

The data we scraped from the internet was extremely noisy. People type whatever comes to their mind. They use weird punctuation marks, unwanted hashtags and lots of emojis. Feeding this dirty text to an algorithm is a very bad idea. That is why data preprocessing became a very strict requirement for our project. We implemented a step by step cleaning pipeline. We wrote python code using regular expressions. This specific code deleted every single emoji and special character. All the capital letters were changed into small letters after that This helps the computer realize that 'Money' and 'money' are exactly the same. The next step was stop word removal. Common words like 'is', 'was', 'the' and 'and' appear thousands of times. But they do not contain any angry or happy sentiment. We removed all these useless words to make the dataset lighter. The very last cleaning step was lemmatization. This grammar technique changes words back to their dictionary form. For example, it converts 'complaining' back to 'complain'.

3.5 Feature Extraction (Vectorization)

Machine learning algorithms require numerical input. Accordingly, we utilized TF-IDF Vectorization to transform the cleaned text into feature vectors.

- **Term Frequency (TF):** Measures the frequency of a word in a document.
- **Inverse Document Frequency (IDF):** Assigns lower weights to common words and higher weights to rare, meaningful keywords.
- **Formula:**

$$TF - IDF = TF \times IDF$$

This approach ensures that domain-specific keywords like "Commission", "Traffic", and "Fraud" are prioritized.

3.6 Exploratory Data Analysis (EDA)

The data structure was analyzed using EDA first. This Exploratory Data Analysis is a key step to check everything in our study We used it to understand the actual structure of the driver dataset. We plotted graphs to check the data distributions and correlations. This visual check helped us find hidden data biases. It also proved the overall data integrity. We completed this exact phase before choosing our preprocessing methods and training the machine models.

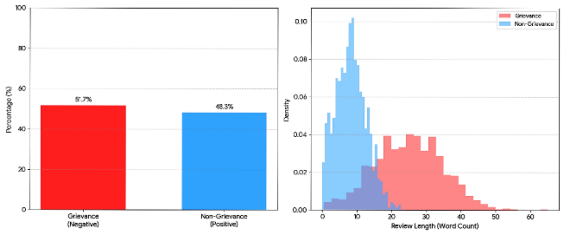

Fig 3.3: Exploratory Data Analysis of Driver Reviews (a) Class Distribution, (b) Review Length Analysis.

a) Class Distribution: The first graph shows a very well balanced data distribution. We have exactly 51.7 percent Negative Grievance records. The Positive Non-Grievance records are 48.3 percent. One single class usually dominates in normal real world datasets. This creates highly biased model predictions. But our dataset has a near equal split. This balance stops the machine learning algorithm from preferring one specific class. It

strongly prevents model overfitting. Our system gets a totally unbiased training process because of this equilibrium. The classifier easily learns the exact patterns of critical driver complaints and normal feedback equally.

b) Review Length Analysis: We compared the review lengths using a simple histogram. We found a very interesting driver behavior here. The red color bars represent grievance reviews. They have a very high total word count. The blue color bars represent positive feedback. Drivers write very long descriptions when they face operational issues like app glitches or payment discrepancies. They want to explain their daily struggle clearly. But the positive reviews are extremely short. Drivers just type words like "Nice" or "Good app". Angry drivers write long paragraphs and this is called verbosity. This specific behavior proves that TF-IDF is the best feature extraction method. TF-IDF easily finds important unique words in long messy texts. So it is the absolute mathematically optimal choice for our specific driver grievance project.

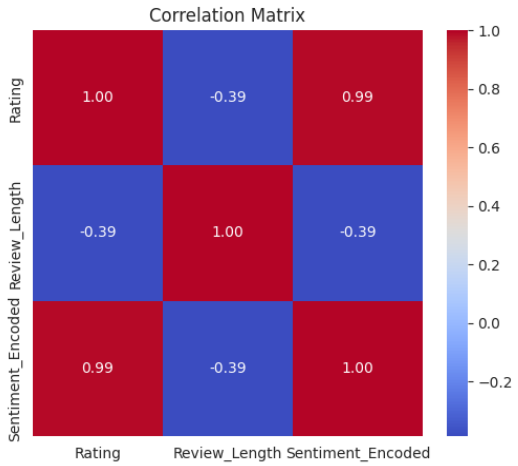

Fig 3.4: Correlation Heatmap displaying the relationship between User Rating, Review Length, and Sentiment.

We did a strict correlation analysis to find hidden patterns in the text data. You can see the results in Fig 3.4. The generated heatmap shows a very strong positive correlation between the user Rating and the Sentiment. The exact correlation value is approximately 0.99 here. This directly confirms that high star ratings always match with positive driver feedback. We also checked the correlation matrix for review length. It showed a moderate negative correlation value of -0.39. This negative value means negative sentiment grievance reviews are much longer and very detailed. Happy drivers do not write long reviews. This clear mathematical insight gave us confidence. It validates our final decision to use TF-IDF for content-based feature extraction. We did not want to rely only on simple metadata like review length.

4. MACHINE LEARNING ALGORITHMS

This study evaluates four distinct algorithms to determine the most effective model for grievance classification.

4.1 Logistic Regression (LR)

Our project uses Logistic Regression specifically for the binary classification part. It works differently from the normal linear regression models because it does not predict continuous values. We applied the Sigmoid Function here to transform the final output. This is a very helpful function because it squashes all our predictions into a probability score. This score always stays between 0 and 1. By looking at this probability, we can easily tell if a driver review should go to Class 0 or Class 1.

Mathematical Formulation:

The core of Logistic Regression is the Sigmoid activation function, denoted as $\sigma(z)$. The hypothesis function $h_\theta(x)$ is defined as:

$$h_\theta(x) = \sigma(\theta^T x) = \frac{1}{1 + e^{-z}}$$

Where:

- (x) represents the input feature vector.
- θ is basically the weights or coefficients we give to each feature.
- e is Euler's number (approximately 2.718).

If $h_\theta(x)$ ≥0.5, the model predicts the positive class; otherwise, it predicts the negative class.

4.2 Multinomial Naive Bayes (MNB)

The Naive Bayes method uses Bayes' Theorem to build different types of classifiers. It works really well when our data has word counts or TF-IDF vectors. This model follows a "naive" assumption, meaning it treats every word as independent from others. Even though this is a very simple idea, MNB gives great results on high-dimensional text data like our driver reviews.

It operates on the "naive" assumption that features (words) are conditionally independent of each other given the class label. Despite this strong assumption, MNB performs exceptionally well on high-dimensional text data.

Mathematical Formulation:

According to Bayes' Theorem, the probability of a class y given a feature vector x is:

$$P(y|x) = \frac{P(x|y) \cdot P(y)}{P(x)}$$

In the Multinomial model, we calculate the posterior probability for each class and predict the class with the highest probability:

$$\arg max_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Where:

- $P(y|x)$ is the posterior probability of the class given the data.
- $P(y)$ is the prior probability of the class.
- $P(x_i|y)$ is the likelihood of feature i appearing in class y

4.3 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a robust supervised learning algorithm used for classification. The primary objective of SVM is to find the optimal Hyperplane in an N-dimensional space that distinctly separates the data points of different classes. SVM maximizes the Margin, which is the distance between the hyperplane and the nearest data points from either class (known as Support Vectors).

To handle non-linear text data effectively, we utilize the RBF (Radial Basis Function) Kernel, which maps input data into a higher-dimensional space where it becomes linearly separable.

Mathematical Concept:

The decision boundary is defined by the hyperplane equation:

$$w^T x + b = 0$$

The algorithm aims to minimize the weight vector $||w||$ subject to the constraint that all data points are correctly classified with a margin ≥1:

$$y_i(w^T x_i + b) \geq 1$$

Where:

- w is the normal vector to the hyperplane.
- b is the bias term.

4.4 Random Forest Classifier (RF)

Random Forest is basically a group of many Decision Trees working together. We call this an ensemble method. Instead of trusting just one tree, which might make mistakes, this algorithm builds a lot of different trees during the training phase. At the end, it takes a majority vote from all these trees to decide the final result. This approach is very good for improving our accuracy and it stops the model from overfitting.

It uses Gini Impurity as a criterion to decide the best split at each node of the trees. A lower Gini Impurity indicates a "purer" node (i.e., data points belong mostly to a single class). **Mathematical Formulation (Gini Impurity):**The Gini Impurity for a dataset containing C classes is calculated as:

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Where:

- p_i is the probability (or proportion) of an item belonging to class i.
- C is the total number of classes.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 03 | March – 2026 DOI:10.55041/ISJEM05749
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

5. RESULTS AND DISCUSSION

5.1 Experimental Setup

We used the following code to set up our machine learning pipeline. It combines TF-IDF Vectorization for extracting features with an SVM classifier that uses an RBF kernel to pick up on driver grievances

Snippet 1:python Implemention of SVM Classifier

Table 3: Accuracy Analysis across Different Data Splits

Split Ratio	Training Size	Testing Size	Accuracy Achieved
60-40	60%	40%	90.00%
70-30	70%	30%	89.90%
80-20	80%	20%	90.08% (Optimal)

To see which setup worked best, we tested a few different train-test split ratios. Looking at the results in our table and Fig 5.1, the 80:20 split was the clear winner.

After testing everything, we saw that training the SVM model with 80% of the data was the best choice, as it reached a 90.08% accuracy. We also looked at the 60:40 and 70:30 splits to compare. While those gave us results around 90.00% and 89.90%, they didn't feel as reliable as the 80:20 split. To be honest, having 80% of the data for training really helped the model pick up the patterns in the driver complaints much better. Honestly, having more training data helped the model learn the specific patterns in driver grievances much better. This confirms that the 80:20 split is the most reliable choice for this study, as shown in the performance trends of Fig 5.1

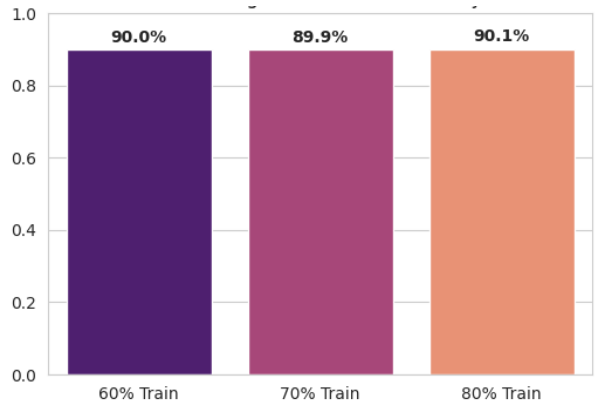

Fig 5.1: Impact of Training Data Split Ratio (60:40, 70:30, 80:20) on Model Accuracy.

```
from sklearn.feature_extraction.text
import TfidfVectorizer

from sklearn.svm import SVC

from sklearn.pipeline import
make_pipeline

# Initialize Pipeline: TF-IDF + SVM
Classifier

model = make_pipeline(

    TfidfVectorizer(stop_words='english',
max_features=5000),

    SVC(kernel='rbf', C=1.0,
probability=True)
)

# Train the Model

model.fit(X_train, y_train)
```

5.2 Performance Metrics

While regular sentiment analysis often marks grievances as 'Negative', our research focuses mainly on Grievance Detection. So, for all our performance checks, we treat 'Grievance' as the Target Class (Positive Class). Here is how we define our terms in this report:

- **True Positives (TP):** Instances where the model correctly identified a Grievance (Actual: Grievance, Predicted: Grievance).
- **True Negatives (TN):** Instances where the model correctly identified General Feedback (Actual: Non-Grievance, Predicted: Non-Grievance).
- **False Positives (FP):** General feedback incorrectly classified as a Grievance (Type I Error).
- **False Negatives (FN):** Critical Grievances incorrectly classified as General Feedback (Type II Error - Missed Detection).

5.2.1 Accuracy

We use Accuracy as a basic measure to check the overall performance of our model. Basically, it is the count of all right predictions (positives and negatives combined) divided by the total number of data points in our set.

Formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 03 | March – 2026 DOI:10.55041/ISJEM05749
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

5.2.2 Precision

Precision is all about how accurate the model's positive predictions are. It calculates what percentage of the identified grievances were actually correct. When a model has high precision, it simply means we aren't getting too many false alarms or wrong flags.

Formula:

$$Precision = \frac{TP}{TP + FP}$$

5.2.3 Recall (Sensitivity)

Recall shows how complete the model is. It measures how many of the actual positive cases the model was able to find. This is very important when missing a grievance is a big problem.

Formula:

$$Recall = \frac{TP}{TP + FN}$$

5.2.4 F1-Score

The F1-Score is the balance between Precision and Recall. Unlike simple accuracy, it looks at both false positives and false negatives. It is very useful for our dataset because it gives a fair score even if the classes are not even.

Formula:

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

5.3 Interpretation of Results

Our tests show that the Support Vector Machine (SVM) classifier is the best performer, hitting a top accuracy of 90.08%. We think SVM did so well because it handles the high-dimensional data from TF-IDF very effectively. As Saleem et al. (2021) mentioned, this method works great for text tasks. Also, SVM is good at creating a clear gap between classes, which keeps the model stable even when the data is a bit sparse.

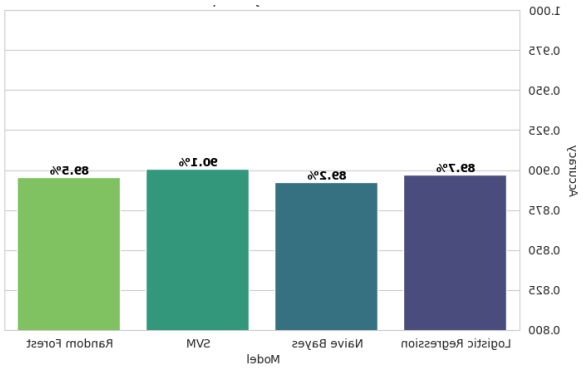

Fig 5.2: Performance Comparison of Various Machine Learning Models

Table3: Performance Comparison of Machine Learning Algorithms

Algorithm	Accuracy	Precision	Recall	F1-Score
SVM	90.08%	86.19%	96.10%	90.87%
RF	89.50%	85.56%	95.40%	90.21%
LR	89.70%	85.75%	95.62%	90.41%
MNB	89.20%	85.27%	95.08%	89.90%

.

Random Forest also did quite well with an accuracy of 89.50%, which matches what Peerbasha et al. (2023) said about how strong ensemble methods can be. Still, it was just a bit behind SVM when it came to picking up the small details in messy driver feedback. Logistic Regression hit 89.70% accuracy, showing it's a solid choice for binary classification, but it just doesn't have the non-linear power that SVM has. Multinomial Naive Bayes had the lowest score at 89.20%. Even though Benjamin et al. (2018) talk about how useful it is for language-based products, its simple "naive" rule probably limited it compared to the complex boundaries SVM can build. In the end, our SVM model was the top performer, getting the highest F1-Score of 90.87% and a really impressive Recall of 96.10% for grievances. This proves it is the best at not missing any real complaints.

5.4 ROC Curve Analysis

To check how well our models can tell classes apart, we did an ROC curve analysis. As shown in Fig 5.3, this curve helps us see the model's ability by comparing the True Positive Rate (Sensitivity) against the False Positive Rate at different settings. The AUC score is basically

the most important part of this analysis. When the value gets closer to 1.0, it shows that the model is doing a great

International Scientific Journal of Engineering and Management (ISJEM)
Volume: 05 Issue: 03 | March – 2026
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI:10.55041/ISJEM05749

job at telling the difference between a 'Grievance' and a 'Non-Grievance' review.

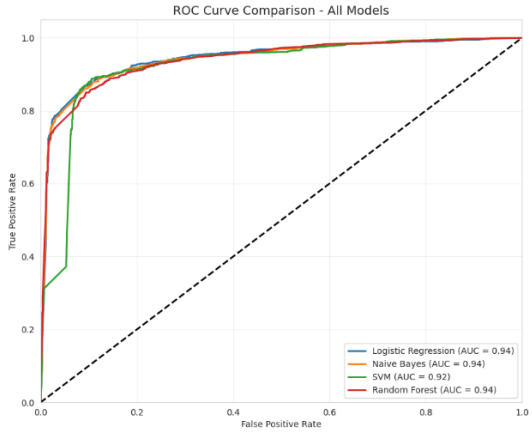

Fig 5.3: Receiver Operating Characteristic (ROC) Curve Analysis comparing AUC scores of SVM, Random Forest, and other classifiers.

The results were also confirmed by our ROC curve analysis. Both Logistic Regression and Random Forest reached the top AUC score of 0.94, showing they are quite good at sorting the data. Even with a slightly lower AUC of 0.92, we decided to stick with SVM as our final model. The main reason is that SVM gave us the best overall accuracy of 90.08%. Also, when we checked the confusion matix, SVM felt much more stable than the other algorithms we tested.

5.5 Confusion Matrix Analysis

To further evaluate the performance of the proposed SVM model, a confusion matrix was generated. This visualization helps in understanding the number of correct and incorrect predictions made by the classifier on the test dataset.

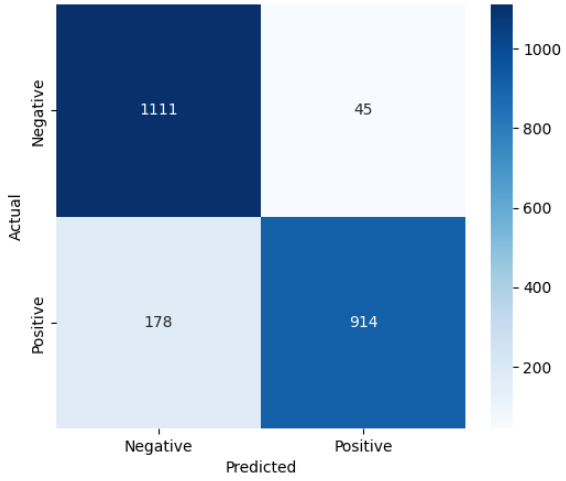

Fig 5.4: Confusion Matrix Heatmap for SVM

We checked the Fig 5.4 confusion matrix to see how our SVM model actually performed on the test set. It gives a really clear picture of where the model got things right and where it tripped up. If you look at the results, the model did a solid job by getting 1111 True Positives and 914 True Negatives. But the best part for us is that it only had 45 False Negatives. This means it almost never misses a real grievance, which is exactly what we were aiming for with this system.

6. CONCLUSION

This research proves that Machine Learning can really help solve the problem of handling driver complaints in the gig economy. Our study shows that using an SVM classifier with TF-IDF is the best way to go for this task. The SVM model worked really well and gave us a top accuracy of 90.08%. Even with all the messy and random driver feedback we had, the model didn't fail and kept a steady performance during our tests. Honestly, it worked way better than the other options like Random Forest or Logistic Regression. This just proves that SVM was the right call for handling this kind of unstructured text data. This shows that SVM is the right fit for handling this kind of text data in our study. By automating how we find and group these grievances, ride-hailing apps can move away from slow, manual support to a much faster system. This allows for real-time tracking of big issues like payment problems or app bugs, which leads to quicker fixes and happier drivers. Solving these problems fast is the key to keeping drivers on the platform and building better trust.

Future Work: Next up, we are thinking about trying some advanced stuff like BERT or LSTM models. Maybe these will pick up on what the drivers really mean in their feedback much better. We are also thinking about adding a voice-to-text option later. It'll be a lot easier for drivers since they can just speak their complaints instead of trying to type while they are busy driving.

7. REFERENCES

1. Surie, A., & Adeel, M. (2018). Ride-hailing apps and the transformation of the driver's work in India Journal of Transport Geography, 73, 56-65. https://doi.org/10.1016/j.jtrangeo.2018.10.013.
2. Home, Prohibition and Excise Department. (2023). Policy Note on Motor Vehicles Acts Administration 2023-2024. Government of Tamil Nadu.
3. Kashyap, R. (2018). Sentiment Analysis of Indian Ride-Sharing Services. 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI). https://doi.org/10.1109/ICACCI.2018.8554766
4. Peerbasha, S., Saleem Raja, A., et al. (2023). Diabetes Prediction using Decision Tree, Random Forest, SVM. Journal of Advanced Applied Scientific Research, 5(4), 42-54.
5. Pandey, S. (2021). Sentiment Analysis of Driver Reviews in the Gig Economy. International Journal of Computer Applications.
6. Scikit-learn Developers. (2023). Logistic Regression User Guide. https://scikit-learn.org/stable/modules/linear_model.html
7. Lakshmanarao, B., et al. (2021). Malicious URL Detection using NLP and Machine Learning. ICSES.
8. Saleem, A., et al. (2023). Weighted ensemble classifier for malicious link detection. Int. Journal of Pervasive Computing.
9. Saleem, A., Vinodini, R., & Kavitha, V. (2021). Lexical features based malicious URL detection. Materials Today: Proceedings, 47(1). https://doi.org/10.1016/j.matpr.2021.05.659
10. Benjamin, B., et al. (2018). Applied Text Analysis with Python. O'Reilly Media. https://www.oreilly.com/library/view/applied-text-analysis/9781491963036/.
11. Géron, A. (2017). Hands-On Machine Learning with Scikit-Learn and TensorFlow. O'Reilly Media.https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/
12. Li, W., & Aletras, N. (2022). Improving Graph-Based Text Representations with Character and Word Level N-grams. arXiv preprint arXiv:2210.05999. https://arxiv.org/abs/2210.05999
13. Jadhav, A., et al. (2022). An Empirical Assessment of Performance of Data Balancing Techniques. Applied Sciences, 12(8), 3928. https://doi.org/10.3390/app12083928
14. Hong, J., et al. (2019). Malicious Domain Names Detection Algorithm Based on Lexical Analysis. IEEE Access, 7, 132641-132649. https://ieeexplore.ieee.org/document/8830336.
15. Selvi, R., et al. (2019). Detection of algorithmically generated malicious domain names. Expert Systems with Applications.
16. Boukhalfa, I., et al. (2020). A new approach for the detection and analysis of phishing in social networks. SNAMS.
17. Joshi, A., Lloyd, L., Westin, P., & Seethapathy, S. (2019). Using Lexical Features for Malicious URL Detection. arXiv preprint arXiv:1910.06277. https://arxiv.org/abs/1910.06277
18. Berina, A., et al. (2017). Machine Learning Techniques for Classification of Diseases. MECO.
19. Ridam, P., et al. (2017). Application of Machine Learning Algorithms on Diabetic Retinopathy. IEEE.
20. Suresh Kumar, P. (2017). Diagnosing Diabetes using Data Mining Techniques. IJSRP.
21. Peerbasha, S., & Mohamed Surputheen, M. (2021). A predictive model to identify possible affected bipolar disorder students using Naïve Bayes, Random Forest, and SVM machine learning techniques. IJCSNS International Journal of Computer Science and Network Security, 21(5), 267-275.
22. Y. M. Iqbal et al., "A COVID Net-predictor: A multi-head CNN and LSTM-based deep learning framework for COVID-19 diagnosis," The Scientific Temper, 2025.
23. Y. M. Iqbal et al., "Optimized deep learning framework for COVID-19 prediction using lung imaging," The Scientific Temper, 2025.

BIOGRAPHIES

Dr. Y. Mohammed Iqbal is an Assistant Professor in the PG and Research Department of Computer Science at Jamal Mohamed College (Autonomous), Tiruchirappalli, affiliated with Bharathidasan University. He holds a Ph.D. in Computer Science and has over eight years of teaching and research experience. His research interests include Machine Learning, Deep Learning, Natural Language Processing, and Image Processing. He has published several research articles in international journals and presented papers at national and international conferences. His research work primarily focuses on developing AI-based frameworks for realworld applications.

A. Abubakkar is a Master of Computer Applications (MCA) student at Jamal Mohamed College (Autonomous), Tiruchirappalli, affiliated with Bharathidasan University. His areas of interest include Artificial Intelligence, Machine Learning, and Data Analytics. He has experience in developing intelligent systems using machine learning techniques and has worked on projects involving **AI-Powered Driver Grievance Detection in Ride-Hailing Services**. His work focuses on applying machine learning models to analyze driver feedback and identify service-related issues to improve platform efficiency and driver satisfaction. He has also gained practical experience in building data-driven applications during his academic projects

Dr. S. Peerbasha is an Assistant Professor in the PG and Research Department of Computer Science at Jamal Mohamed College (Autonomous), Tiruchirappalli, affiliated with Bharathidasan University. He holds a Ph.D. in Computer Science and over Seventeen years of teaching and research experience. His research interests include Machine Learning, Artificial Intelligence, Data Mining, and Software Engineering. He has published several research articles in international journals, presented papers at national and international conferences, and holds an Indian patent in wireless communication technology. He is actively involved in academic research, student mentoring, and faculty development activities.

Dr. M. Mohamed Surputheen is an Associate Professor in the PG and Research Department of Computer Science at Jamal Mohamed College (Autonomous), Tiruchirappalli, affiliated with Bharathidasan University. He holds a Ph.D. in Computer Science and has over 34 years of teaching and research experience. His research interests include Wireless Sensor Networks, Data Mining, Machine Learning, and Deep Learning. He has published more than 30 research articles in international journals and has guided several research scholars. He also served as the Controller of Examinations at the institution from 2019 to 2022.

Dr. M. Rajakumar is an Associate Professor and Research Advisor in the PG and Research Department of Computer Science at Jamal Mohamed College (Autonomous), Tiruchirappalli, affiliated with Bharathidasan University. He holds a Ph.D. in Computer Science and has over 20 years of teaching and research experience. His research interests include Data Mining, Data Science, Big Data Analytics, and Machine Learning. He has supervised several M.Phil. and Ph.D. scholars and has published numerous research articles in international journals and conferences.