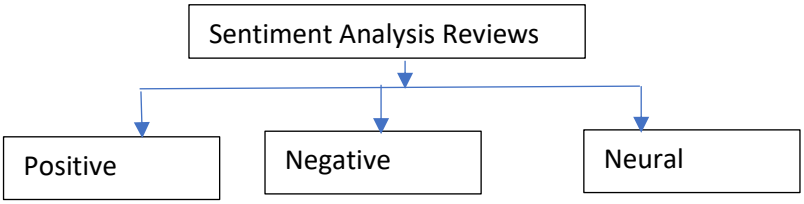
3.2 Sentiment Analysis Project
Trip Advisor Hotel Reviews

Objective

The objective of this sentiment analysis project is to understand how customers feel about TripAdvisor hotel reviews (positive, negative, or neutral). By analyzing the sentiment of reviews, we aim to gain insights into customer satisfaction and dissatisfaction, thereby understanding the pain points of customers and identifying features that contribute to their experiences.

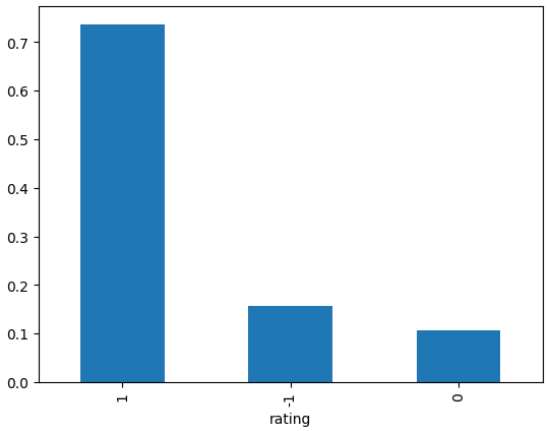


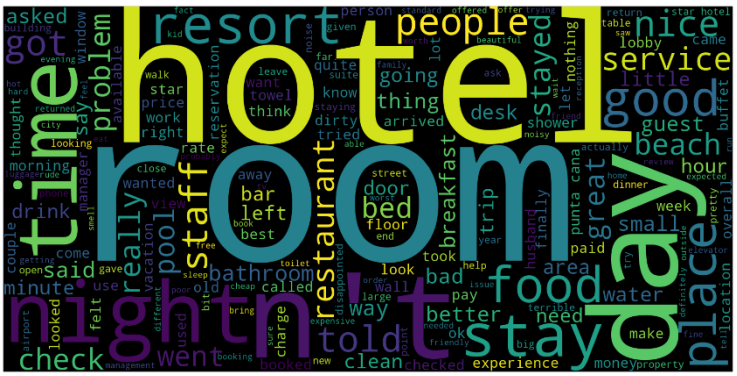
1. Data Preprocessing

Text Cleaning:- Removed Special characters, punctuation, and stopwords are removed from the review text to reduce noise.

Text Normalization:- Lemmatization is performed to reduce words to their base forms, aiding in standardizing the text.

2. Data Visualization: (Matplotlib and wordcloud library used to get this plot)



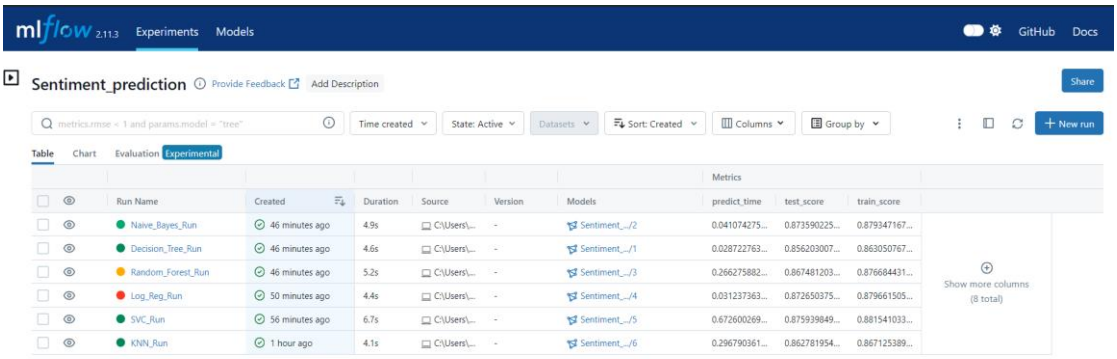


Based on the above visualization we can take Data-driven insites like:

1. **Positive Sentiment Dominance**: The majority of feedback is positive, indicating high satisfaction.
2. **Room Quality Enhancement**: Use feedback to elevate room quality and guest satisfaction.
3. **Customized Dining Experience**: Tailor food offerings to match guest preferences for better dining experiences.
4. **Efficient Operations**: Streamline check-in/out processes based on insights to minimize wait times and enhance efficiency.

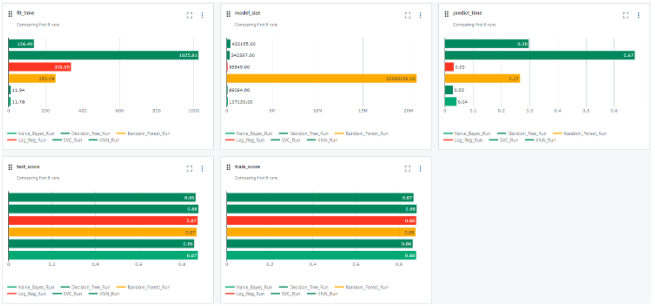
3. MLFlow: Unified Platform for Experiment Tracking and Model Registry

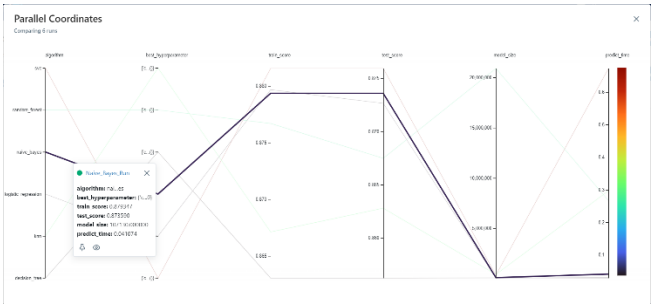
1. Experiment Tracking



1. **Create Experiments**: Establish experiments in MLflow for each model variant.
2. **Define Naming Conventions**: Maintain clear naming conventions for experiments and runs.
3. **Log Parameters**: Record model hyperparameters and configurations.
4. **Log Metrics**: Track evaluation metrics such as accuracy for each model run.
5. **Track Artifacts**: Optionally, log artifacts like model binaries or visualizations.
6. **Compare Results**: Utilize MLflow to compare model performances and select the best.
7. **Iterate and Optimize**: Iterate on models based on insights, optimizing performance

2. Metrics and Hyper parameter Visual Analysis





Model	Accuracy	Efficiency	Scalability	Interpretability	Assumptions / Characteristics
Naive Bayes	83.7%	Shortest training time (0.83 seconds)	Relatively small model size	Simple and interpretable	Relies on feature independence
Decision Tree	77.34%	Easy to understand and visualize	Can overfit	Model size grows with larger datasets	Tuning can mitigate overfitting
Logistic Regression	87.0%	Provides insights into feature importance	May not scale well to very large datasets	Longest training time	Regularization techniques prevent overfitting
Random Forest	86.4%	Ensemble learning, combines multiple trees	Longer training time compared to some models	Largest model size, higher memory requirements	Significant impact of hyperparameters

Based on the trade-offs between prediction time, fit time, model size, test score, and train score, **Logistic Regression** emerges as the best choice for the given data. Here's why:

1. **Prediction Time**: While Logistic Regression may not have the fastest prediction time compared to Naive Bayes, it still offers reasonable prediction times for most datasets.
2. **Fit Time**: Logistic Regression typically has shorter fit times compared to more complex models like Random Forest, making it efficient for training on moderate-sized datasets.
3. **Model Size**: Logistic Regression models have relatively small memory footprints compared to ensemble methods like Random Forest.

4. **Test Score and Train Score**: Logistic Regression achieves competitive test scores while maintaining a good balance with train scores, indicating reasonable generalization performance without significant overfitting.

4. Deploying a Flask Web Application on AWS Cloud

Step 1: Set up AWS Account and Resources

1. Create an AWS Account
2. Launch EC2 Instance

Step 2: Prepare Your Flask Application

1. Import the necessary modules (`flask`, `Flask`, `render_template`, `request`, and `joblib`).
2. Initialize a Flask application instance.
3. Define a route for the home page (`'/'`) where users can input reviews. The function `home()` renders the `home.html` template.
4. Define a route for prediction (`'/prediction'`) where users can submit their review for sentiment analysis. This route accepts both GET and POST requests. In case of a POST request:
    - Retrieve the review from the form.
    - Load the pre-trained model (`naive_bayes.pkl`).
    - Predict the sentiment of the review.
    - Render the `output.html` template with the prediction result.
5. Run the Flask application in debug mode on `0.0.0.0`.

Step3: HTML Code Structure:

- Define the document type and language.
- Include metadata in the head section, such as character set and viewport configuration.
- Style the body with center alignment, background color, and font family.
- Create a container div for the main content with specified styling.
- Display the title and form on the home page.
- Display the prediction result on the output page.

Step 3: Configure Your EC2 Instance

1. SSH into Your EC2 Instance
2. Install Required Packages
3. Copy Flask Application

Step 4: Access Your Flask Application

1. Open Web Browser
2. Enter EC2 Instance's Public IP Address or Domain Name