REPORT

By

Kushal(IMT2021035)
Chokshi(IMT2021012)
Raghunadh(IMT2021042)

Introduction

The data you have pertains to whether a bus arrives on schedule or experiences a delay. Your task is to
predict the number "0" when the bus is on time and "1" when it is running behind schedule. Features in
the given data set:

1. Index-index
2. Bus_ID- The identification or number assigned to each bus currently in service.
3. DepartureTime- Departure time of the bus in minutes.
4. Journey_Time- Total time taken in minutes for the bus to reach its destination.
5. Bus_Operator- The company that owns/operates a given bus.
6. Departure_Bus_Stop- Location where bus starts journey
7. Arrival_Bus_Stop- This refers to the bus stop where the bus is expected to arrive.
8. Day- which day it is.
9. Target- 0 if the bus is on time to its destination. 1 if the bus is delayed.

Preprocessing and EDA

Removing Duplicate Rows

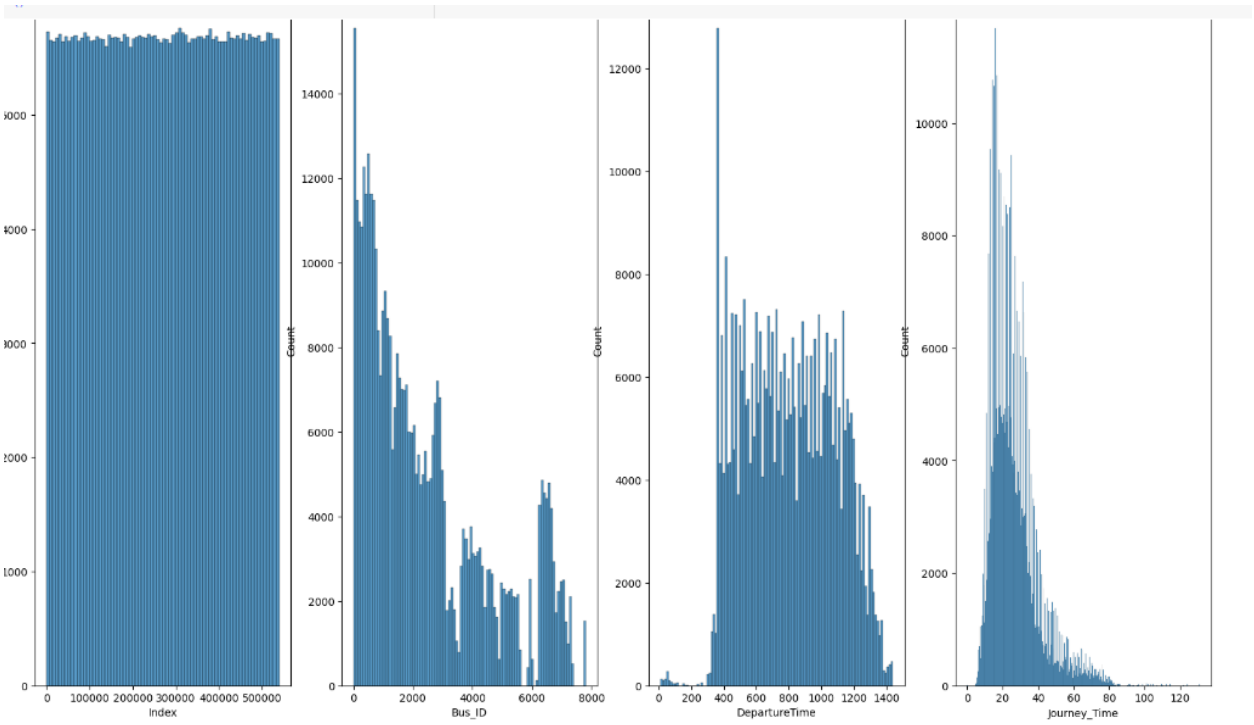
We checked the number of duplicate rows in the dataset and removed them.

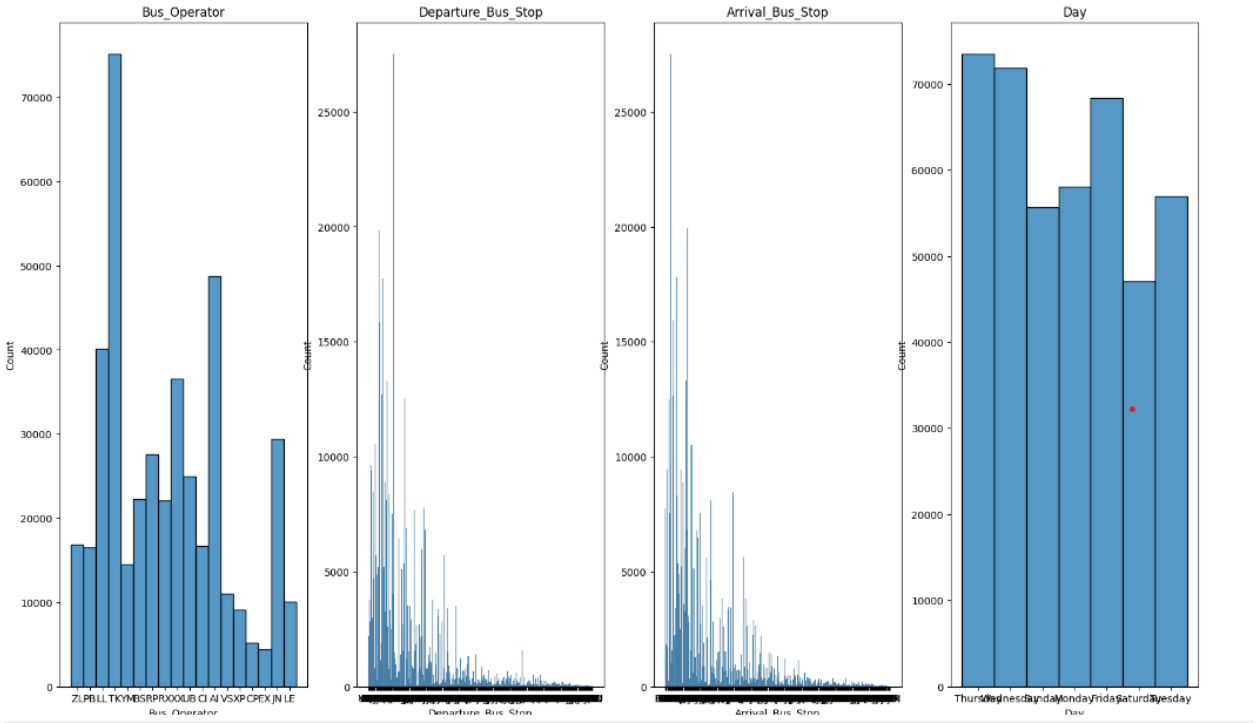
Dealing with null values

We have to remove null values because they are redundant and are not needed.
There are null values in 2 columns of the dataset. They are Journey_Time and Day.
For **Journey_Time** column,
We observed that journey times of a bus with the same Bus_ID are nearly the same. We grouped data
based on **Bus_ID** and found the mean of each group. We replaced null values of that column by
mean of that group. Even in test_data, we did the same to replace null values of this column.

For Day column,
We observed that most buses having same Bus_ID have one a particular day's frequency more
than others. We grouped data based on Bus_ID and replaced null values with mode of that Data.

Hist Plots for each column to observe the distribution:





Encoding

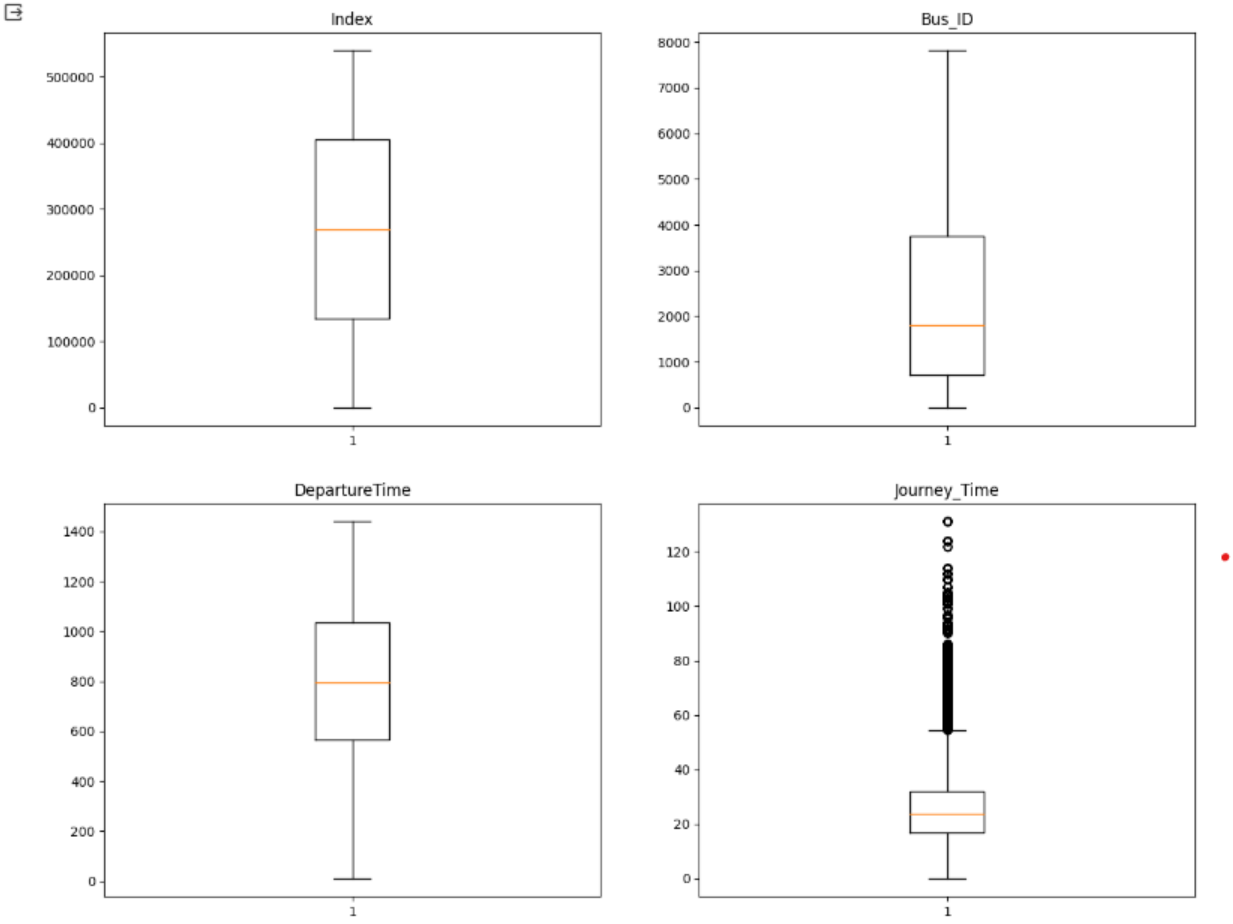
We use encoding to change any categorical columns to numerical data as many machine learning algorithms run on numerical data only.

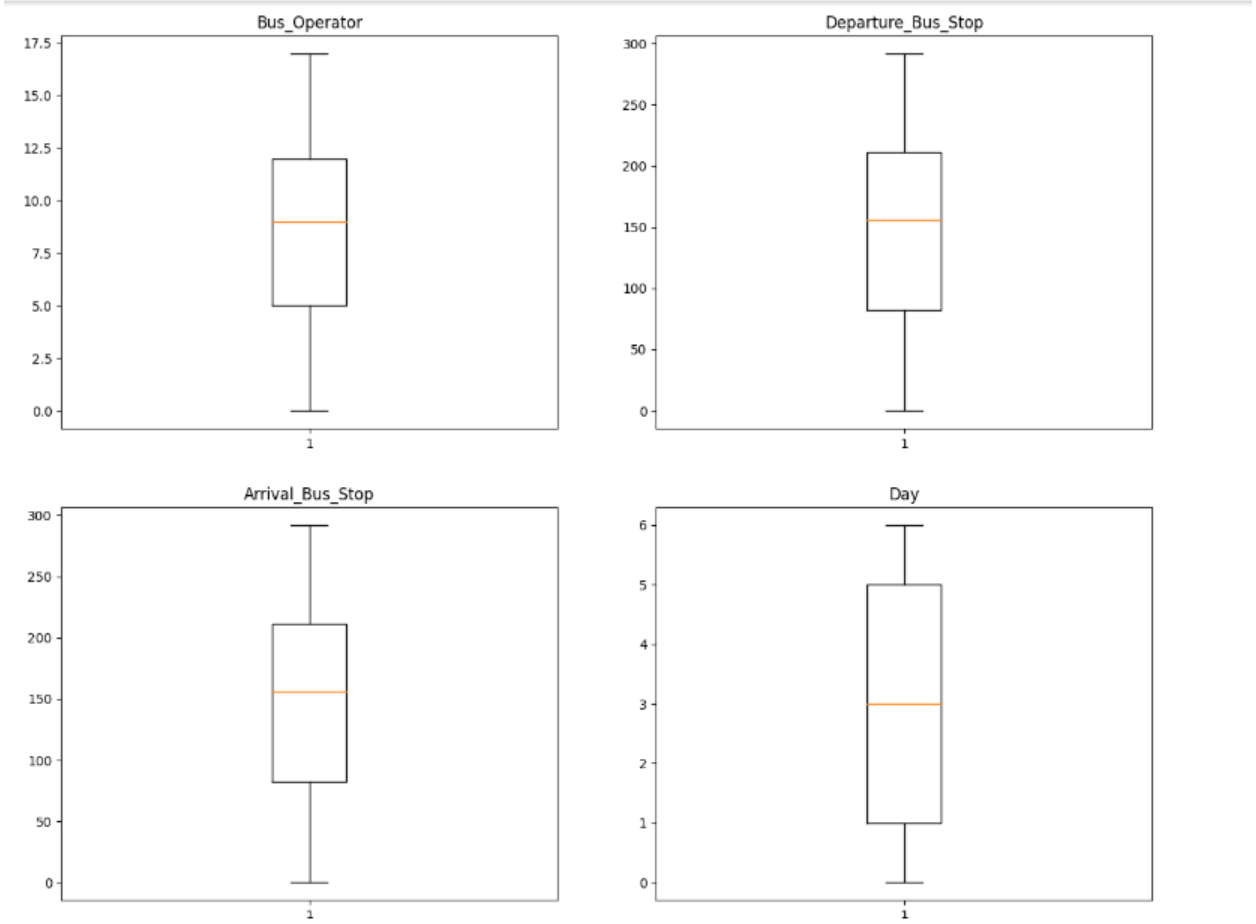
Label encoding

We concatenated training and test data, performed label encoding and then separated them.
We did label encoding to all columns having string values.
The following columns required label encoding:

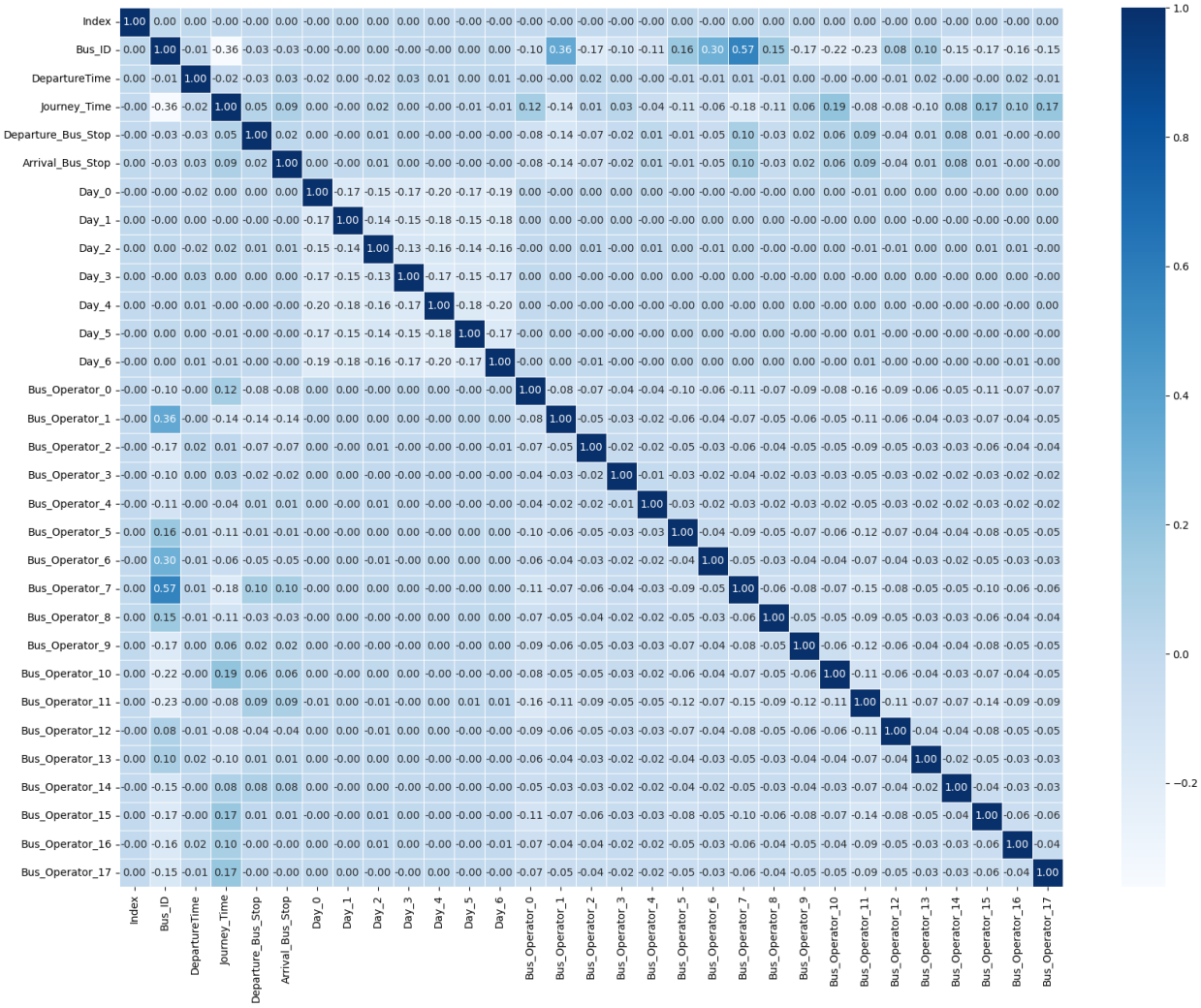
- Arrival_Bus_Stop
- Departure_Bus_Stop

Plotting boxplots to detect outliers in each feature. Traveler Type and Checking Service need outlier removal:


Index
500000
400000
300000
200000
100000
0
1
Bus_ID
8000
7000
6000
5000
4000
3000
2000
1000
0
1
DepartureTime
1400
1200
1000
800
600
400
200
0
1
Journey_Time
120
100
80
60
40
20
0
1



Plotting the heatmap to get the correlation between the columns:



One hot encoding

We did one hot encoding to the column Day because there are only 7 unique values and
Bus_Operator column as it has only 16 unique rows.

Non neural ensembles

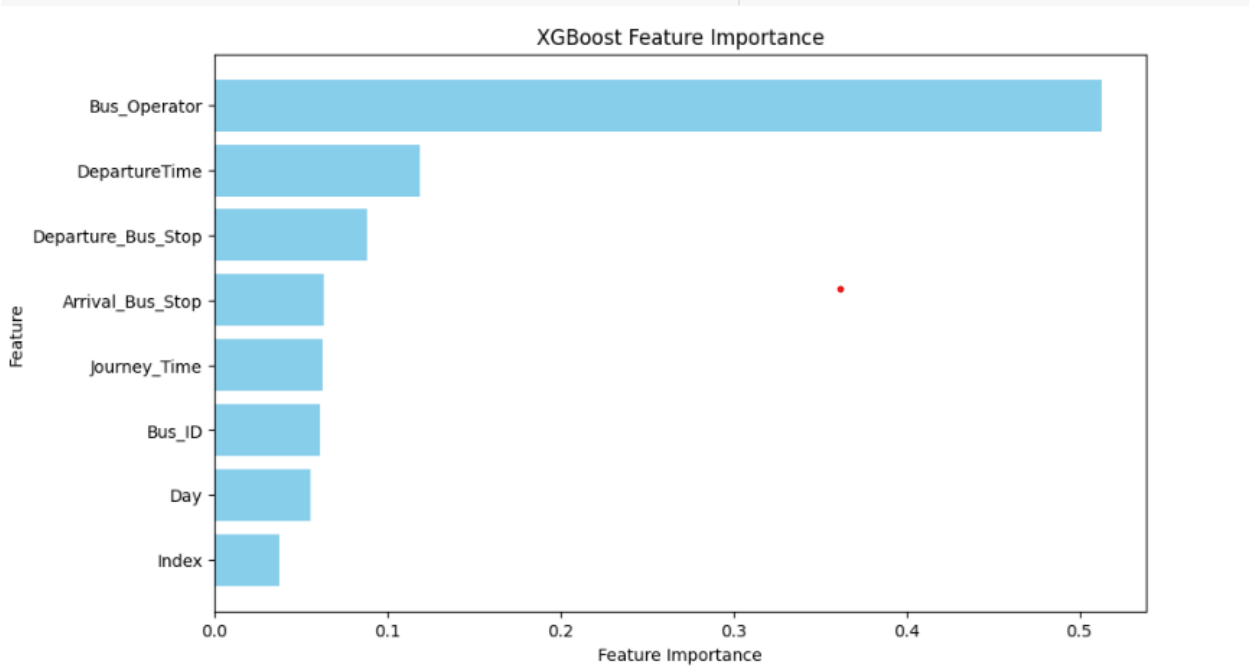
We used the following models to train data and predict on test data. The accuracy scores of
Each model provided insights into how well the models have classified data.

• Random Forest and Grid Search:

We used random forest classification and grid search. The accuracy we obtained is 63.78%. So,
We decided to try boosting techniques as we got less accuracy.

• XG Boost with Grid Search:

At last, we did XG Boost and achieved an accuracy of 66.78%. We also used grid search
and did hyperparameter tuning to get this result.
We got the best result with XG Boost with grid search method after doing hyper-parameter
tuning.



Making prediction on test data

So, we trained data using XGBoost and predicted values on test data.

SVM

Hyperparameter Tuning for C:

We did hyperparameter tuning for the regularization parameter C. Utilizing the Support Vector Classification model, a randomized search is conducted over a set of predefined candidate values for C. The hyperparameter search aims to find the value that maximizes the F1-weighted score, and the best C is extracted from the results.

Hyperparameter Tuning for Kernel:

Following the determination of the optimal C, the code proceeds to hyperparameter tuning for the kernel. The SVC model is configured with the previously identified best C, and a randomized search is performed over possible kernel choices—'linear', 'rbf', and 'sigmoid'. The kernel with the highest F1-weighted score is considered the best, providing insight into the most suitable kernel function for the SVM model.

Hyperparameter Tuning for Gamma:

In the final section, the code addresses hyperparameter tuning for gamma. Leveraging the best C and kernel identified in the previous steps, a randomized search explores different values for gamma. The goal is to pinpoint the gamma value that optimizes the F1-weighted score, contributing to a more fine-tuned and effective SVM model.

Final SVM Model Creation:

The concluding steps involve constructing the final SVM model with the optimal hyperparameters, including C, kernel, and gamma. The SVC model is instantiated with these best values, and the model is fitted to the training data. This comprehensive approach ensures the SVM model is fine-tuned for optimal performance based on the identified hyperparameter values. At last, we did predictions using the best model.

Neural Networks

Neural Network Architecture and Compilation:

We used a neural network using TensorFlow's Keras API, with a sequential model structure. It consists of a flattened input layer, followed by a dense layer with six units and a rectified linear unit (ReLU) activation function. While the initial implementation includes dropout layers for regularization, they are currently commented out. Subsequently, two more dense layers with two and one unit(s) respectively are included, with the final layer utilizing a sigmoid activation for binary classification. The model is compiled using the Adam optimizer, binary cross-entropy loss function, and accuracy as the evaluation metric.

Hyperparameter Configuration:

. We tried using different numbers of layers and different numbers of nodes in them. We even tuned the learning rate. We adjusted the hyperparameters as needed.

Model Inference and Probability Prediction:

Following model compilation, the code performs inference on standardized test data . It predicts class probabilities for each sample in the test set, representing the likelihood of belonging to the positive class.

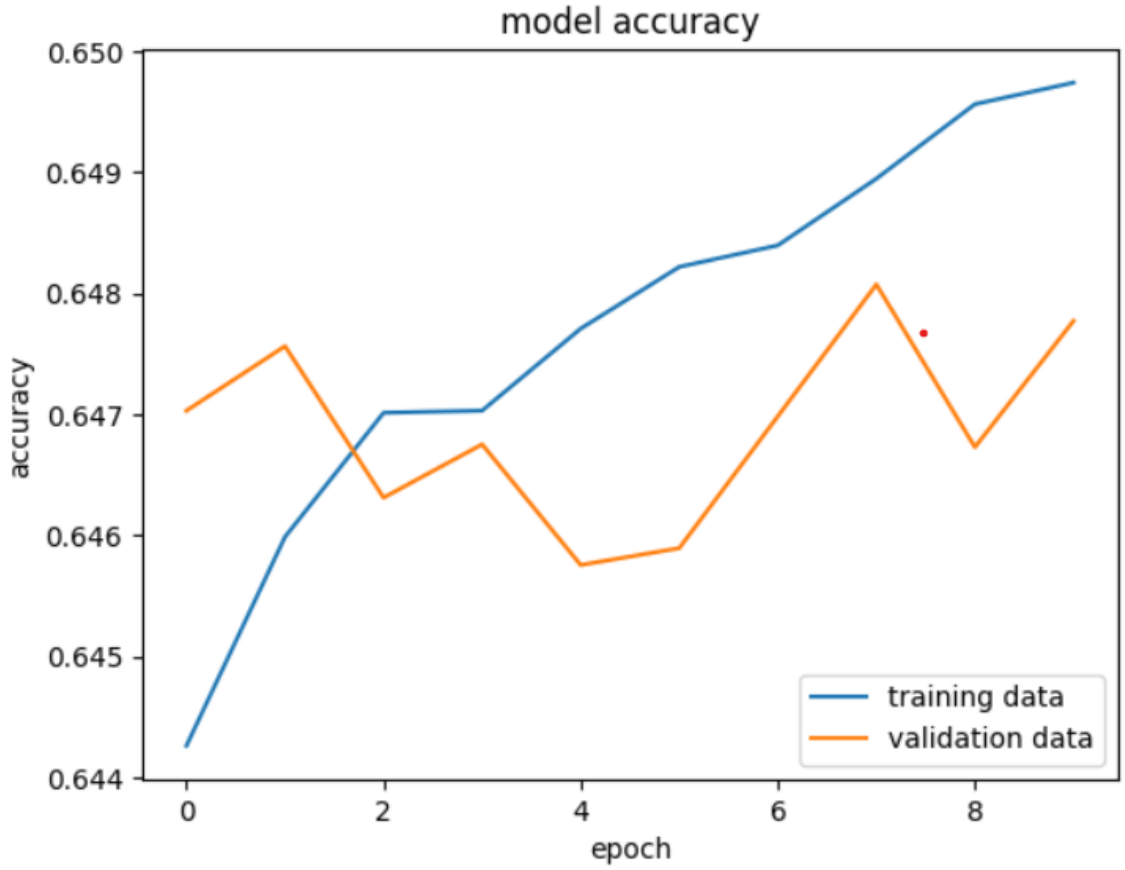
Converting Probabilities to Class Labels:

The method for obtaining class labels in the provided code involves converting the predicted probabilities into discrete class assignments. We achieved it by selecting the class with the highest predicted probability for each sample.

We generated graphs depicting accuracy and loss to assess and analyze the optimal parameters.

Visualizing accuracy and loss:

A line plot to visualize how the training and validation accuracy of a machine learning model change over different epochs. Monitoring these metrics can provide insights into the model's performance and help identify issues such as overfitting or underfitting.



By visualizing the loss over epochs for both training and validation data, you can gain insights into how well the model is learning and whether it is overfitting or underfitting. Monitoring loss is a common practice in training machine learning models to assess their performance during training.