Intelligent Notification System: Identifying Opportune Moments For Mobile Phone Alerts

Kul Gaurav

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Data Science)

Supervisor: Dr. Gavin Doherty

September 2020

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Kul Gaurav

September 4, 2020

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Kul Gaurav

September 4, 2020

Acknowledgments

I would like to express my sincere gratitude to my dissertation advisor, Dr. Gavin Doherty, for his valuable and constructive suggestions throughout the research work. He supervised me in every aspect of the dissertation, from the ethical approval to directing me in the right direction whenever I needed his guidance. I am also thankful to the Computer Science and Statistics Department of Trinity College Dublin to provide me with the environment and support to work on my thesis even during the pandemic.

I would also like to thank my second reader, Dr. Tim Savage, for his support and valuable feedback, which helped me immensely improve my work.

I would like to extend my gratitude to my family and friends for all the motivation and support during this wonderful journey.

Kul Gaurav

University of Dublin, Trinity College
September 2020

Intelligent Notification System: Identifying Opportune Moments For Mobile Phone Alerts

Kul Gaurav, Master of Science in Computer Science

University of Dublin, Trinity College, 2020

Supervisor: Dr. Gavin Doherty

With the advent of modern smartphones and easily accessible internet, push notifications flood our devices daily. We interact with a large number of notifications each day on our mobile devices, coming from various sources like social media, instant messaging, news, games, and other applications installed. These notifications are not always relevant to our current state and also often received untimely. In this research, we collected data from mobile devices of volunteers and analyzed to learn about opportune moments for a notification. The development and deployment of the android application for data collection were according to our university's ethical research guidelines. The Covid-19 created a unique situation and usage pattern for mobile devices. We saw the contrast from similar previous studies because of the new normal of staying inside and working from home. Exploratory Data Analysis and Statistical Methods performed for identifying features and patterns in the collected data. Supervised machine learning was applied to test the hypotheses around classifying a moment as opportune to deliver a notification. The XGBoost classifier gave the best results for the classification of a moment as opportune or not. Exact time prediction to deliver a notification was inefficient, suggesting the requirement of additional context and content features for a better model.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Our day starts and ends with the interaction with mobile phones hence making it an integral part of our daily lives. We interact with various contents throughout the day. According to a report by Deloitte [1], Irish people check their phones around 50 times a day, most of which to engage with the alert notification. Figure 1.1 shows the usage distribution. From social media updates to application reminding to drink water, we get numerous push notifications each day, which often confound us. Often the notifications we receive, create disturbances in our routine task or add no value to our daily lives [2, 3]. We try to ignore these notifications and open them in our free time.

Nevertheless, these unwanted notifications already disturbed and diverted our attention from our work by generating irrelevant thoughts and mind-wandering. Many companies deliver push-based notifications for marketing and advertisement purposes, but untimely delivery does not add any value to their goal as not serving the purpose of people's engagement. Delivery of notification at an appropriate time can help not just the phone user by increasing productivity but also the companies generating push notifications by making the audience interact with the notification.

Notifications on a mobile device create an additional display, usually an icon on top of the screen, and alerts the user by a sound. The device may also create vibrations depending on the settings by the user. The notification drawer registers each alert in

the list with the icon of the application. The icon, when tapped, expands to display the details about the notification. An application can receive either a unicast notification which is unique to the user or a broadcast notification that every user of that application receives. In the backend server, a WebSocket or another Back-end-as-a-Service (BaaS) [4] provider triggers the notification that a mobile client device receives and displays.

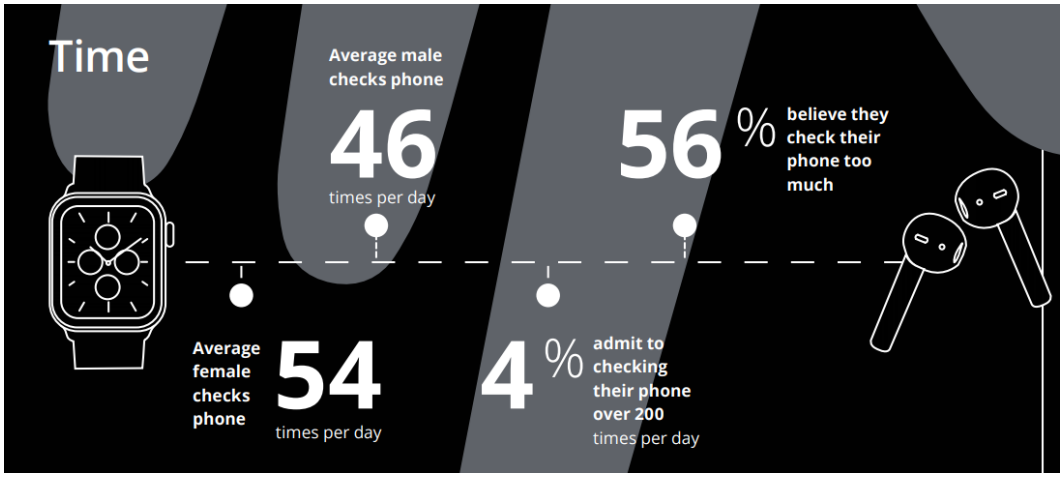

Figure 1.1: Phone Usage Distribution in Ireland [1]

Multiple factors determine if a mobile phone user will open a notification in the desired time range. Data Analytics and machine learning can help understand user behavior and build models that may help develop intelligence in the delivery system of notification. Such a system can benefit mobile phone users as well as marketing companies.

This study's core objective is to model the conventional notification system of mobile phones and identify the opportune moment for the delivery of a notification. The goal is to reduce the diversion created by notifications by identifying whether current factors may lead to the right moment for interacting with the notification quickly and hence increasing the engagement rate.

For developing such a system, we face constraints of open dataset availability. In this study, we explore various features that play a vital role in interacting with a notification at a given time. We also propose a framework for collecting user data with security and privacy considerations to develop and deploy the data collection system in real-time.

1.2 Challenges

Designing and developing a smart notification system is onerous, as it involves many modular tasks. The process starts with data collection from various sensors of the mobile device and saving the data for future use. Analysis of context and content of notification to building the intelligent notification system is another principal task. These tasks carefully need monitoring and best practices for security and privacy concerns. Another challenge is in following iterative model design as there is no data to build a machine learning model initially. This issue is known as the bootstrap problem [5].

Geolocations are an essential feature to predict the opportune moment for a notification. However, an unprecedented situation like Covid-19 can break the model by making the geolocations irrelevant in a lockdown situation and creating high multicollinearity between features.

1.3 Motivation

Since the mobile phone users receive a large number of notifications each day, they may miss crucial alerts in torrent. From a business point of view, if the notifications arrive at an inappropriate time, the end-user may not engage as expected or even unregister from the subscription out of annoyance. The notification system with intelligence can help reduce the disturbance in the ongoing task of end-user. Though content and sensor data together give a good model for the system, machine learning algorithms in the domain using limited contexts and features are still under research.

Broadly the intelligent notification system can have elements as in figure 1.2. After the alerts reach the mobile gateway, a classifier can accept the notification if it is relevant to the user; further, the system looks for an opportune moment to post the notification. In this study, we do not consider the notification's relevance as that involves content analysis of the notification. Our work focuses on the opportune moment classifiers and regressors.

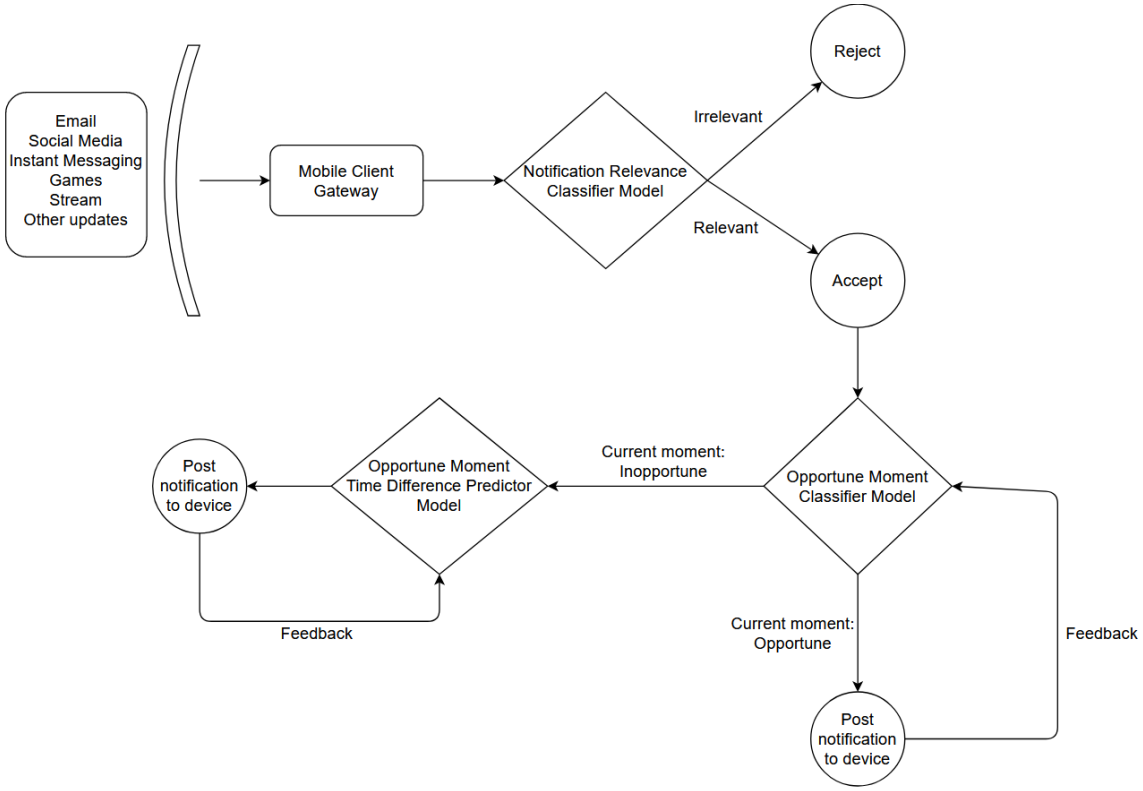

Figure 1.2: Overview of Intelligent Notification System

1.4 Research Question

The research question that this thesis is seeking to answer is:

" Can we predict the opportune moment for a notification arriving at a mobile device by using limited context features without reading the notification content? "

Prediction of the opportune moment is of the following three types:

1. Classifying current moment as opportune or non-opportune.
2. Multi class classification of next opportune moment for a notification from one of the following class:
   (a) Instantly ($\leq$ 30 seconds)

(b) Early ($\leq$ 5 minutes)

(c) Delayed ($\leq$ 30 minutes)

(d) Very Late (anything after 30 minutes)

3. Time difference prediction to obtain the next opportune moment from current time.

Chapter 2

Related Works

Digital reforms around the world have captivated the researchers to study the impacts and consequences of growing mobile phone usage. The industry, as well as academia, is diving deep to understand mobile devices' usage and help enhance the output and productivity of end-user. In the current section, we will discuss some of the past work and recent research output in notification management.

2.1 Effects of Notification Alerts

People mostly check their phones in response to notifications alert [1]. [6] defines a notification as "a visual cue, auditory signal, or haptic alert generated by an application or service that relays information to a user outside her current focus of attention". Phone users do not engage with all of the notifications received, and the content and sender play a vital role in receptivity [7, 8]. Dismissal of notifications is usually for game invites, predictive suggestions, and promotional emails [9]. Nowadays, most of the applications provide the option to mute notifications generated by them. But people prefer to uninstall the app when they receive irrelevant and unwanted notifications [10].

[11], in their research study, establishes that social pressure and expectations in personal communication are the principal ground for interacting with notifications within minutes. Further, an increased number of alerts from email and social networks correlates with negative emotions and stress, while personal communication and instant messaging applications create elevated feelings of being connected. Since few collected

features are non-normal distributed, the study uses non-parametric methods of Friedman test, Wilcoxon rank-sum test, and Spearman's Rho from inferential statistics. The authors analyzed data collected for seven days, which was insufficient to build any model for solving the problem.

2.2 Understanding Features for Prediction

In [12], researchers set up an SMS server and RSS feed to generate pseudo content and capture response. Properties of the notifications' content seem to be more stable for predicting a suitable delivery time for alerts. Recent advancements in natural language processing can further enhance the feature extraction from the content. In the future, if mobile devices have more resources to run sophisticated machine learning models, then the content can add a lot of value in notification management. But reading and analyzing the content of notification over a cloud server is not ideal concerning security and privacy.

Hyungik in [13] presents a framework to notify about missed or rejected calls on time at a recognizable moment. The study utilizes current user activity to build an incremental Naïve Bayes model. There is additional rule-based filtering that enhances the model. Depending on the intensity of time suitability on the scale of 0 to 4, the framework creates a notification to alert. The study shows that users' physical activity can be a vital factor in identifying opportune moments. Authors have considered a limited number of features for the research, but the framework can accommodate new features.

For recording the engagement in recommended contents, [14] asks the volunteers to answer a questionnaire based on their mood. A pseudo content notification follows the questionnaire that can be ignored or attempted. Authors discover that communication is one of the most important features for identifying an opportune moment. Unfortunately, Apple does not provide any API to read call logs in iOS; only Android OS can use the communication logs to build a model. The authors measure the performance of the model by considering all the data at once, and no individual comparison is present.

In [15], the researchers establish the relationship between the end of an episode and an opportune moment where a voice call or reading SMS constitutes an episode. These naturally occurring breakpoints provide contexts to determine the moment of

interruption. As ANOVA would require an equal number of repeated measurements, authors use linear mixed models (LMM) to perform data analysis. LMM helped to understand the participant's unique differences, and if it impacts on the result. The study finds that the time just after finishing a task is the best context as an opportune moment.

2.3 Model Building Approaches

[16] presents a smartphone library that can work over applications to provide intelligence to notifications by classifying a time as convenient for delivery or not. The study also involves understanding the sentiment and engagement features for a user. Authors use WEKA and MOA toolkits to build the interruptibility model. Performing the computations on the local device itself is beneficial for data usage, energy consumption, and privacy concerns. While these toolkits are good to test a hypothesis, they may not be beneficial in production and deployments. Libraries like scikit-learn [17] can provide more model choices and better memory management.

Rule-based learning is incorporated by [9] to create filters on notifications. The researchers also provide an option for end-users to attach custom filters based on the words in the notification message, time of receptivity, or generating application. The model gives extremely high accuracy because of the user's custom filtering. The proposed solution relies on content analysis of notifications to predict the usefulness, which can be a privacy concern.

[18] addresses the problem of identifying the right device to deliver a notification in a multi-device environment. The features for the model are similar to any other notification management system. The authors use Pearson's Chi-squared test and the Gini Index to understand the significance between dependent and independent variables. Ensemble learning models perform the best to classify the suitable device. People nowadays often use more than two devices to separate their profession and personal work. In that case, the model may get more complicated, but the methodology can be helpful for feature engineering and statistical analysis.

2.4 Applications

Businesses spend a massive amount of their budget on digital advertisements. Users often disable the notifications or uninstall the applications that generate alerts at an irrelevant time. [19] focuses on identifying ad scheduling, ad targeting, ad delivery and posting. But end-users are also concerned about the resources consumption by apps running in the background to generate behavioral targeting notifications [16, 10].

Smart delivery of notifications can find applications in various fields ranging from a business point of view to personal well-being. [20] assesses the feasibility of a mobile phone application to record recurrent self-reports for identifying antenatal mood and depression. The notification to log the users' record was prompted on random but in a fixed time interval. Identification of good delivery time can enhance the engagement with applications for mental health screening. [21] thoroughly investigates mobile phones' significance and challenges in the Ecological Momentary Assessment (EMA). Notifications can capture the users' attention for the EMA engagement, which establishes a higher longitudinal and ecological validity. Suitable delivery time for these notifications can enhance the engagement and capture of appropriate sampling data.

Chapter 3

Methodology

3.1 Data Collection

Around 86% of mobile phones in the world run on Android [22]. The Linux kernel-based Android OS gives more access to resources and sensor data than iOS, which is the second-largest OS in use after Android. To conduct the study, we collected the data by deploying an android application on the volunteers' phone. The development and distribution of the app followed the university's guidelines for security and privacy measures.

We developed an android application with the target API level 29 (Android 10) and minimum support API level 21 (Android 5). The purpose of the app was to passively collect different features when a user interacts with a notification, and save it to the cloud server.

After the development, we advertised on social media, asking people to volunteer for the study by installing the application. The app was available to download from the android play store and GitHub repository. Volunteers installed the app after accepting the consent form conditions.

We collected the dataset for the study between March 31st, 2020, and April 30th, 2020. The data collection framework setup followed the ethical guidelines and got the approval from the Research Ethics Committee (REC) of the university.

3.1.1 Ethics Guidelines

We asked android phone users of age greater than 18 years to help us collect data points from their phone usage. The app asked to accept the terms on the consent form before the installation. Volunteers were free to withdraw at any point of time by emailing us, and we securely removed their data as per the request. The location of the server stack for our study was in Europe to follow data sovereignty. The study provided debriefing arrangements for the volunteers on our application installed on their phone or via an email request.

The data anonymization followed the General Data Protection Regulation (GDPR) guidelines. Hashed ids replaced all the user email ids that we collected to identify each user uniquely and provide authentication. We did not receive any other Personally Identifiable Information (PII) from the volunteers.

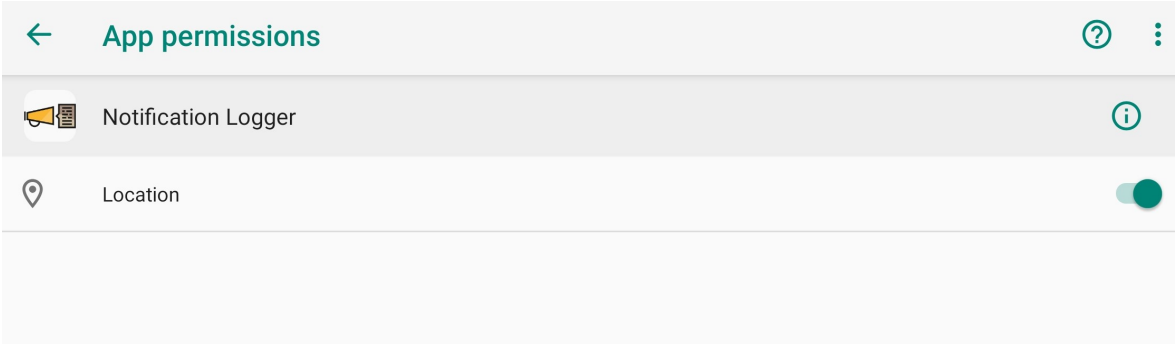

Figure 3.1: Location Permission Toggle Screen

The location and current user activity data was optional for the volunteer to share. One could toggle the location permissions at any time from the app, as shown in figure 3.1. The current activity collection was available in one of the two versions of the application, and the user could install either. Our study did not involve any analysis on the location coordinates and was collected only to build features based on the relative distance between two points.

3.1.2 Application System Design

The app always runs in the background after installation and restart itself if killed during clearing system resources or reboot. When a notification is received or removed on the mobile phone, it collects the snapshot of various sensors and status data at that moment. The local memory of the phone [23] maintains the collected data until the app connects to the Google cloud platform to save the data on the cloud.

A time-based job scheduler (the cron job) ran every day on our cloud server for around four weeks until users had the application installed. The figure 3.2 shows the data flow diagram for the data capture process. This scheduled command anonymized the data by replacing the email id with custom user id and downloaded it to our local server.

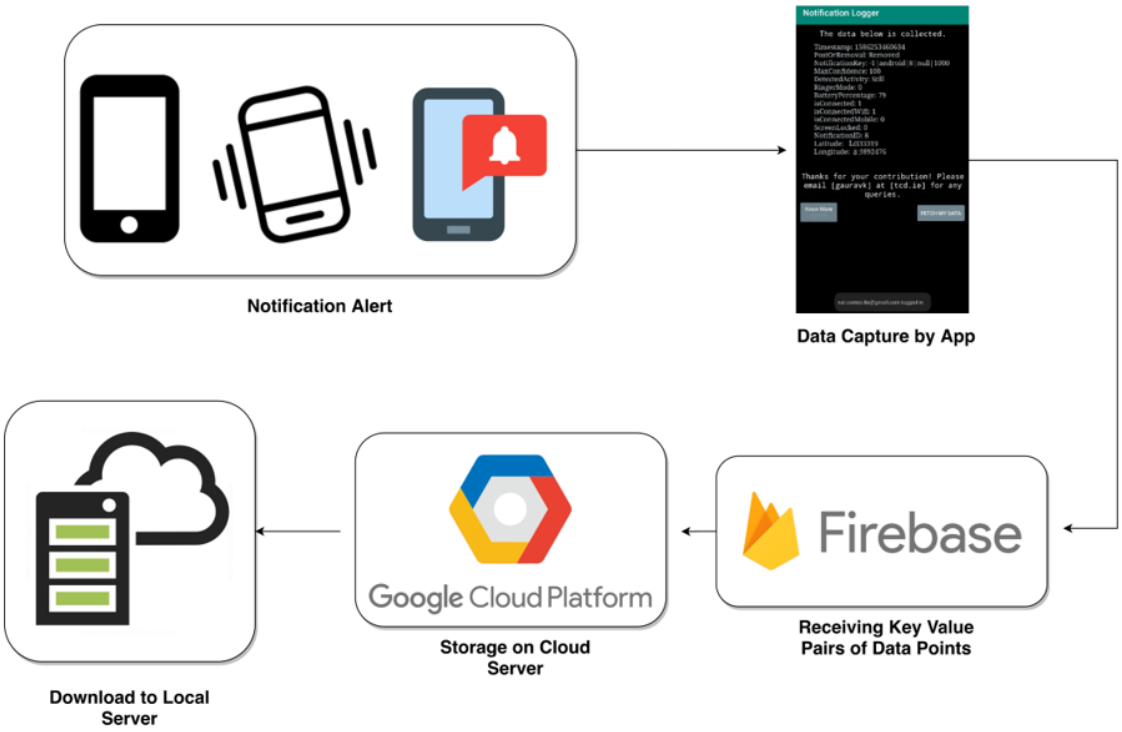

Figure 3.2: Data Collection Flow Chart

Once the data collection was complete, volunteers received a broadcast message notification asking to uninstall the application. Following it, we closed the cloud server access from any client device and removed the app from the Google play store. We used Firebase as the backend-as-a-service for authentication, cloud server connection, and analytics. The user gets an option on the app interface to download data saved from their device besides getting every instance displayed on the app screen, as shown in the figure 3.3.

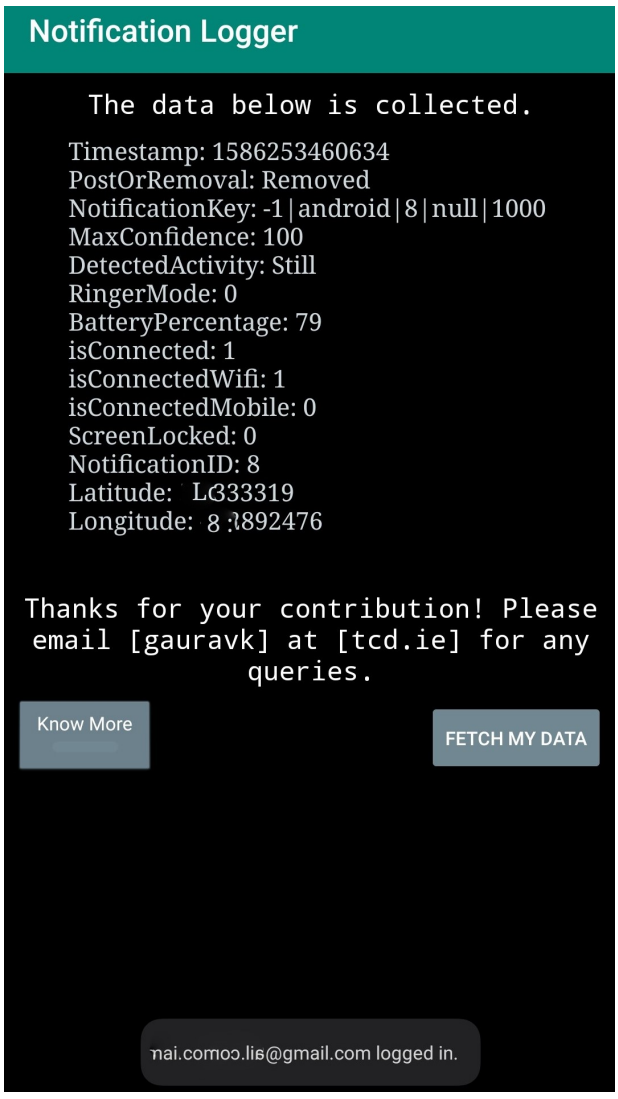

Figure 3.3: User Interface of the Data Collection Application

Our app collects the following data on posting or removal of a notification:

Field	Data Type	Description	Access API
Timestamp	Long	The UNIX epoch timestamp	Notification Broadcast Receiver
Post or Removal	String	If the current notification is posted or removed	Notification Broadcast Receiver
Notification Key	String	A unique key generated by Android OS for each notification, this field also contains the name of package receiving the notification	Notification Broadcast Receiver
Notification ID	Integer	ID assigned to notification by OS	Notification Broadcast Receiver
Detected Activity	String	Current activity of the user. This can be one of the following value: 1. Still 2. On Foot 3. On Bicycle 4. In-Vehicle 5. Running 6. Tilting 7. Walking 8. Unknown	Google Activity Recognition [24]

Maximum Confidence	Integer	A value between 0 to 100. Represents the confidence of current detected activity prediction	Google Activity Recognition
Ringer Mode	Boolean	If the phone can create sound alerts	Media Audio Manager
Battery Percentage	Integer	A value between 0 to 100. The current battery level	OS Battery Manager
Network Connectivity	Boolean	If the phone is connected to the internet	Connectivity Manager
WiFi Connectivity	Boolean	If the phone is accessing a WiFi	Connectivity Manager
Mobile Data Connectivity	Boolean	If mobile data is active	Telephony Manager
Screen Locked	Boolean	If the phone is locked and the screen light is off	OS Keyguard Manager
Latitude	Double	Geolocation latitude of the user	Android GSM Location
Longitude	Double	Geolocation longitude of the user	Android GSM Location

Table 3.1: Fields of Collected Data

3.2 Classification Algorithms

3.2.1 Logistic Regression

This supervised learning classification algorithm runs on the logistic function. A type of sigmoid function, the logistic function is the S-shaped curve (figure 3.4) of the following equation.

$$f(x) = \frac{l}{1 + e^{-}k(x - x_0)} \tag{3.1}$$

where,

x_0 = the middle value of the function

l = the maximum value of curve

k = the logistic growth rate

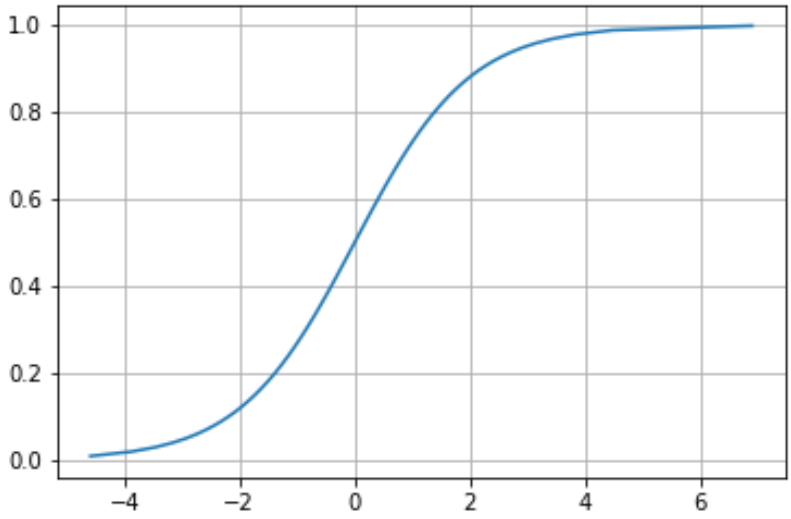

Figure 3.4: Sigmoid Function

The binary logistic regression has the following assumptions:

1. The dependent variables for each observation are always one of the two classes of the output variable.
2. There are no outliers and possibly no misclassification in the training data.
3. There is very little or no multicollinearity among the independent variables. [25] proposes that a correlation coefficient of less than 0.9 meets the condition.
4. The minimal sample size for training should be ten times the number of independent variables.

The algorithm may give vague results or even fail to converge when the above conditions break.
Classification algorithms try to create a decision boundary between two classes to separate each other. For logistic regression, this boundary is calculated using conditional probability. The method calculates the likelihood for each instance belonging to either class of the target variables. The value obtained can lie in the range from negative infinity to positive infinity. Hence, we need the sigmoid function to normalize on the lower scale of zero and one as probabilities.

Loss function (also known as the cost function) is the measure for the performance of a predictive model by quantifying the difference between predicted and expected value. One of the standard loss functions is Mean Squared Error (MSE), but the sigmoid function leads to a non-convex curve when applied for logistic regression that makes the identification of the global minimum difficult.

Hence logistic regression uses logistic loss as the loss function. It can extend for multi-class classification by one-vs-rest (OvR) technique or changing the loss function to cross-entropy loss.

3.2.2 Random Forest Classifier

The ensemble methods blend multiple models to form a possibly better model by involving a tradeoff between extra computation and poor learning by individual models. Random forests are one such method consisting of many different decision trees. The ground truth for the performance is that trees protect each other from their individual errors, and hence the uncorrelated models add positively to the combined model.

Random forest uses the bagging technique with feature randomness while building each tree. Bagging (Bootstrap aggregating) helps reduce the variance and hence lower the chances of overfitting [26]. Voting from each model combines to get the final result where each model has equal weight, as shown in the figure 3.5.

Each tree follows the following rules for construction:

- Given N as the number of instances in the training set, each tree's training set holds the random N samples with replacement.
- For a value k, constant for the whole forest and I as the number of input variables,

k variables are selected randomly from I to find the best split. The constant k is adjusted such that $k << I$.

- The tree does not prune and is grown to the broadest possible size.

Forest can minimize the error by following two procedures:

1. Decreasing the correlation between each tree.
2. Decreasing the error rate of each tree. The lower the error rate for a tree, the stronger it is as a classifier.

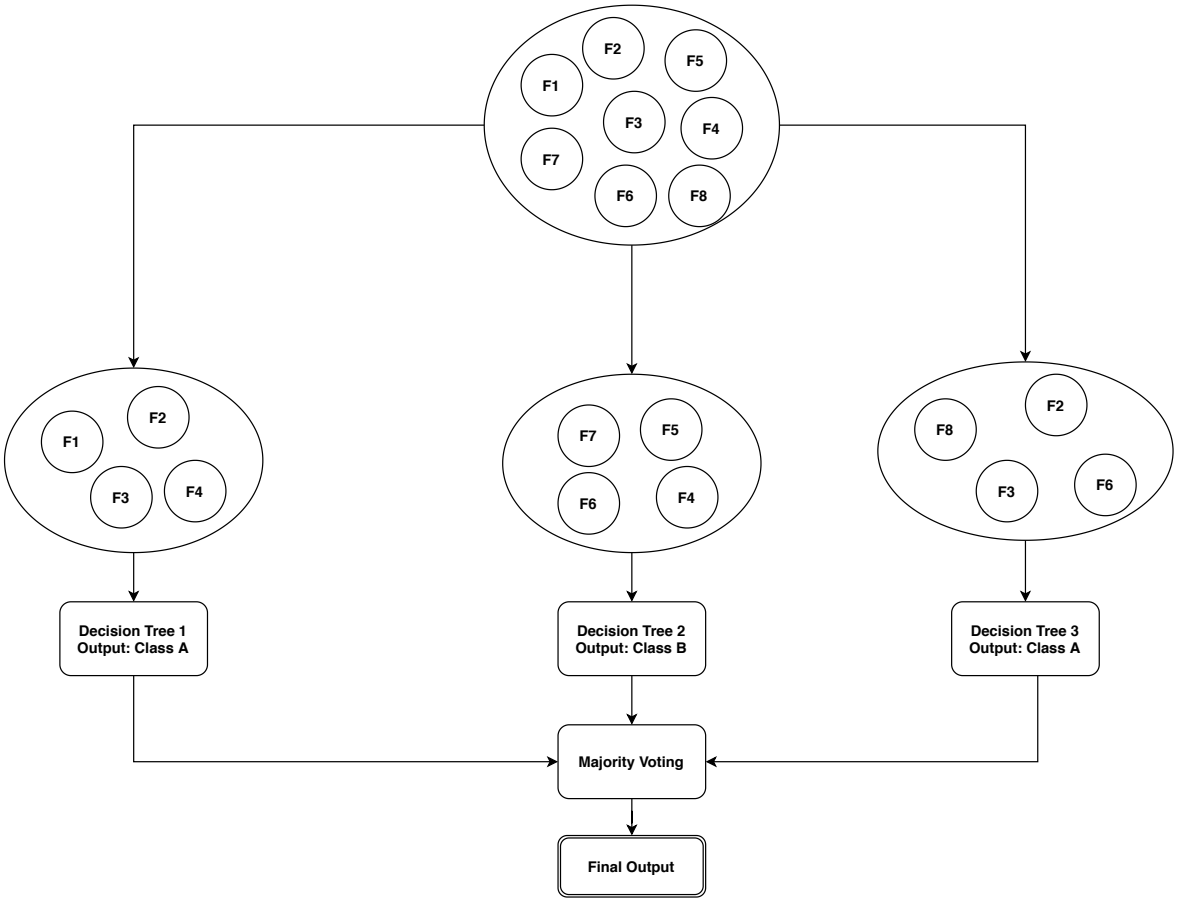

Figure 3.5: Generation of Random Forest

3.2.3 Support Vector Classifier

Support Vector Classifier (SVC) finds the hyperplane (decision surface) that can separate different classes of dependent variables. In general, for a given set of instances, there can be an infinite number of such planes. The SVC finds the optimal one to maximize the margin between the closest samples (forming support vectors) from two different classes.

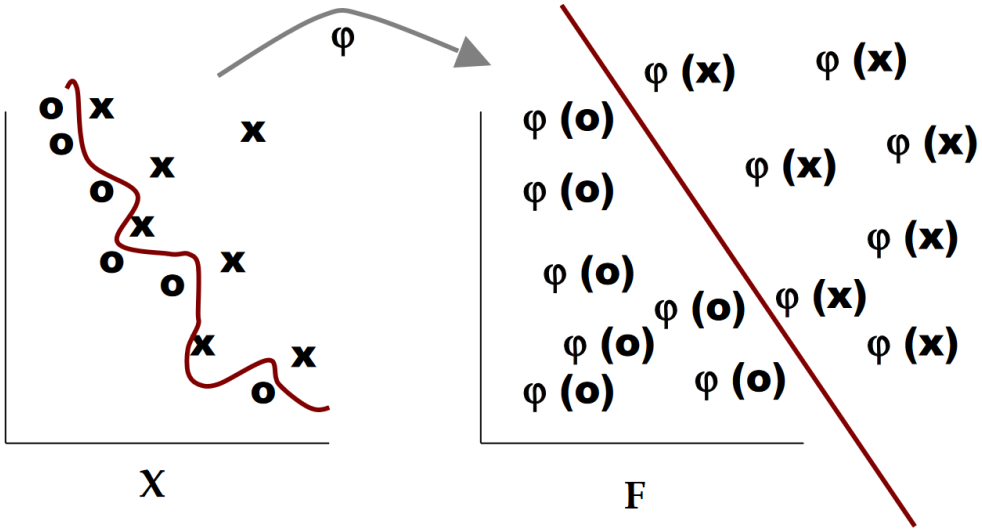

Figure 3.6: Feature Transformation in Support Vector Classifier

A line can separate the classes when we have two dimensions, but we need hyperplanes for higher dimensions. Here the number of dimensions is equal to the number of independent features. The algorithm transforms the data to a higher dimension if there is no linearly separating plane, as shown in figure 3.6. The optimization of maximizing the width between support vectors on a hyperplane is the core objective of SVC. The objective function reduces to a quadratic optimization problem constrained with inequalities with a single global minimum that can be solved using the Lagrange multiplier method.

The algorithm uses a technique known as the kernel trick to increase the dimension space and enhance the performance of the model. Common kernels are:

- Linear Kernel

$$K(\mathbf{f}, \mathbf{f}') = \sum(\mathbf{f} * \mathbf{f}') \tag{3.2}$$

- Polynomial Kernel

$$K(\mathbf{f}, \mathbf{f}') = \left(\mathbf{f}^\top \mathbf{f}' + c\right)^d \tag{3.3}$$

- Radial Basis Function Kernel

$$K\left(\mathbf{f}, \mathbf{f}'\right) = \exp\left(-\frac{\|\mathbf{f} - \mathbf{f}'\|^2}{2c^2}\right) \tag{3.4}$$

In the above equations, f and f' are vectors of features, c is a free parameter, and d is the degree of polynomials. Figure 3.7 shows the working of a Gaussian kernel.

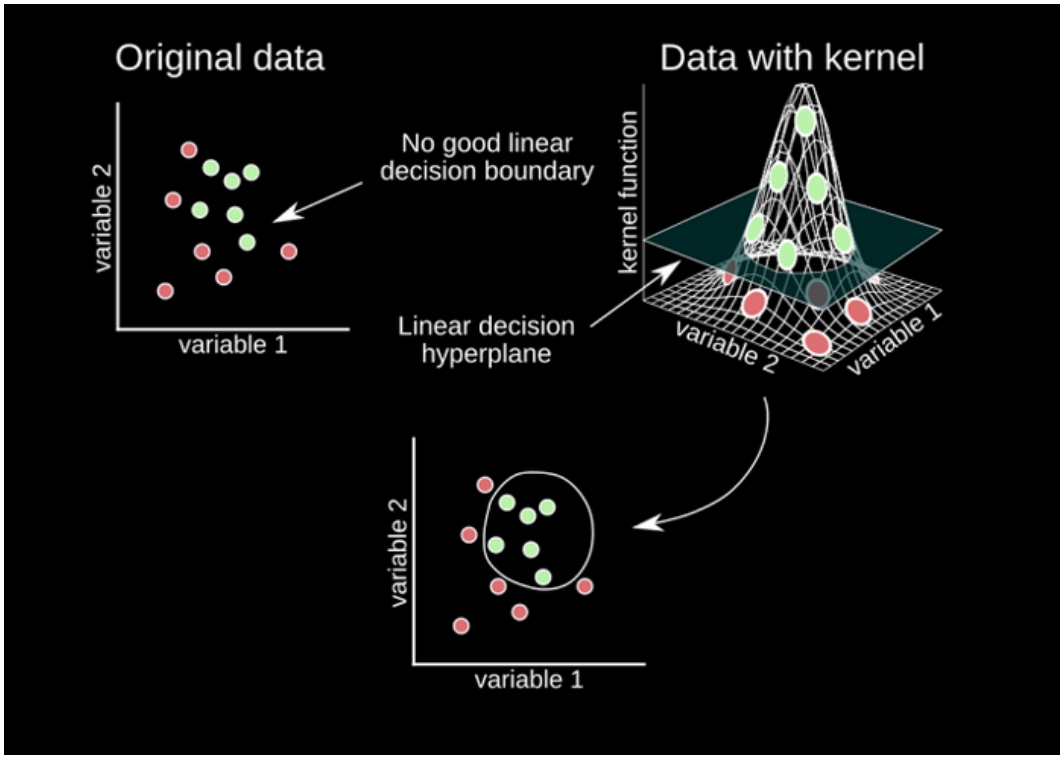

Figure 3.7: Transformation Using Kernel Function

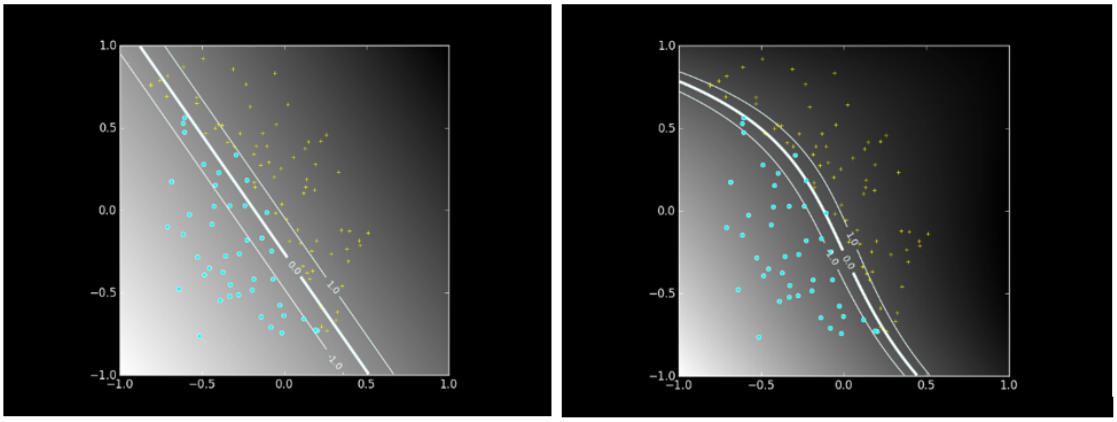

Figure 3.8: Effect of Increasing Degree of the Polynomial Kernel

The variables of the kernel are available as hyperparameters that can be tuned to achieve better performance. Most of the SVC parameters do the trade-off between margin maximization between support vectors and error in the classification. Similarly, kernel parameters like the degree of the polynomial affect the decision boundary. A large value for the degree maps to the complicated relationship between features and helps separate using high dimensional hyperplanes. Figure 3.8 shows one such effect.

3.3 Regression Models

3.3.1 Linear Regression

One of the most extensive approaches to model the correlation between dependent variables and the independent variable is linear regression. A regression line fits the change in predictors in an attempt to identify a linear relationship. Following are the major assumptions for the model:

1. Existence of a linear relationship between dependent and independent variables.
2. Normal distribution of the data.
3. Very little or no multicollinearity between independent features.

4. No auto-correlation between independent features.

5. Equal variance across the residuals on either side of the regression line.

The model has the following hypothesis function:

$$h_\theta(x) = \theta_0 + \theta_1 x \tag{3.5}$$

where,

θ_0 and θ_1 are parameters for the regression line.

The cost function which we minimize over the parameters is:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right)^2 \tag{3.6}$$

The bias and the variance are two fundamental properties of estimators for any model. The assumptions made by the model to improve the learning time constitute the bias while the amount by which the target function change when the training data change is variance. A low value of variance and bias is desirable to accept a regression line. Figure 3.9 shows the three cases of fitting approaches around the original data.

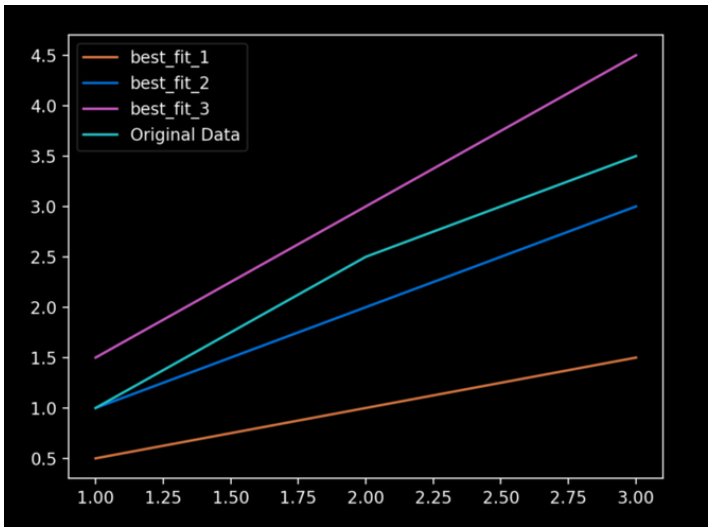

Figure 3.9: Fitting of Regression Lines

For the minimization, linear regression generally uses a gradient descent algorithm that runs till we obtain an acceptable convergence. This iterative approach to find a local minima moves steps that are proportionate to the negative of the slope of the function at the current point. The learning rate determines the step size. Figure 3.10 shows the convergence using gradient descent.

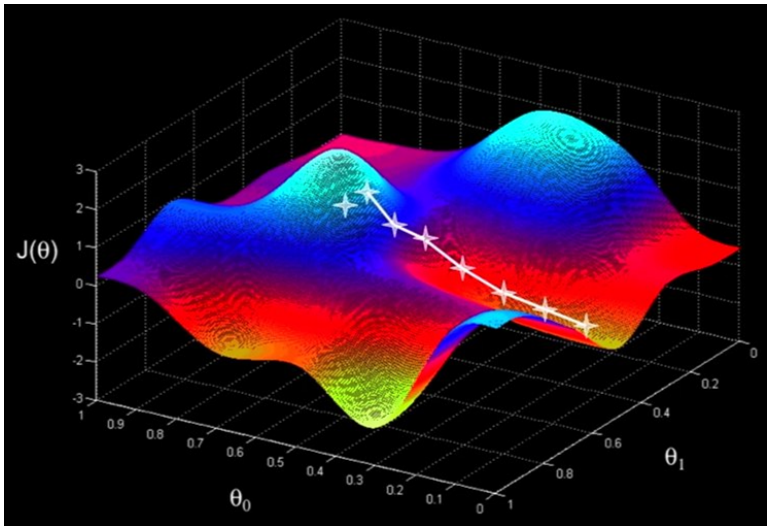

Figure 3.10: Gradient Descent Convergence

3.3.2 Lasso Regression

Linear regression fails to give good results when the features show a substantial correlation between each other. LASSO stands for Least Absolute Selection Shrinkage Operator, and the method minimizes the effect of few predictors by turning their corresponding coefficients smaller or even zero [27]. Making the coefficients exactly zero is one technique of feature selection, and lasso provides it out of the box. The cost function to minimize changes as follows:

$$J\left(\theta_0, \theta_1\right) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{3.7}$$

where,

λ is the tuning parameter and β is the constraint for each coefficients.

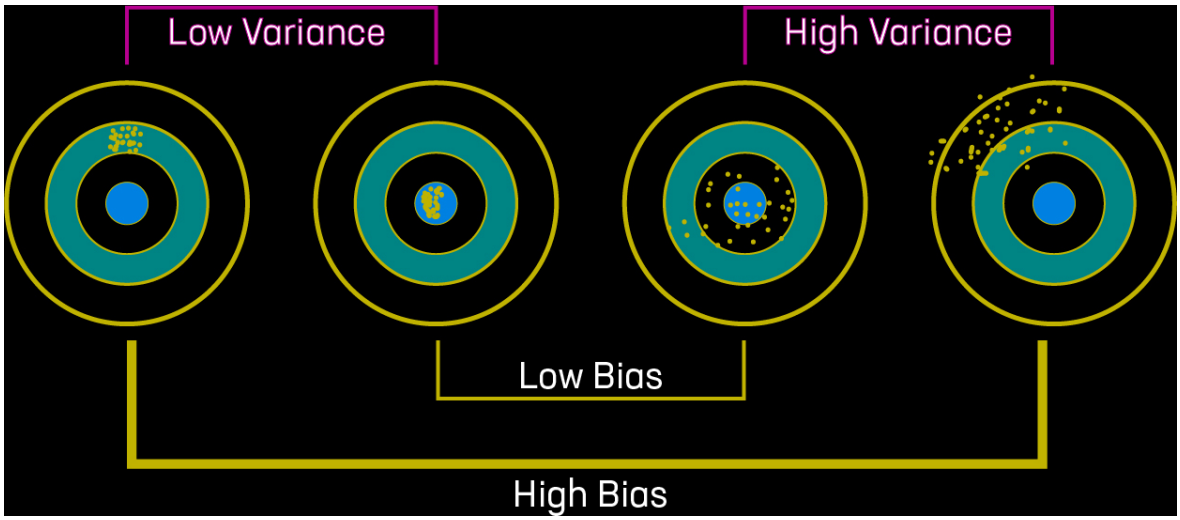

Figure 3.11: Variance Bias Trade-off Visualization

The tuning parameter controls the strength of the penalty on coefficients. With the increase in the tuning parameter, the model's bias increases, whereas a decrease leads to an increase in the variance. It is not possible to ignore the relationship between bias and variance as decreasing one will increase the other. 3.11 shows the association between them. Hence, a suitable value of the tuning parameter helps to improve the model with a trade-off.

The algorithm works well when we have a small number of significant features. In figure 3.12, we evaluate the regression for our data set by iterating the model training for various lambda values. We can also identify the best tuning parameter by analyzing the regression coefficients and performing cross-validation. The figure 3.13 shows the elbow effect for choosing a good lambda with cross-validation.

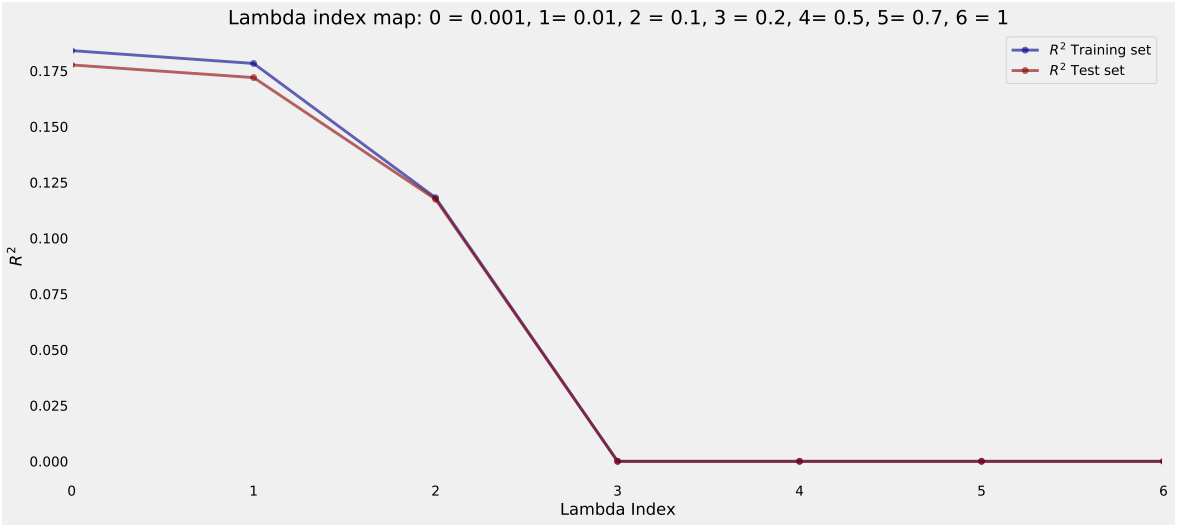

Figure 3.12: Lasso Regression Evaluation With Varying Lambda

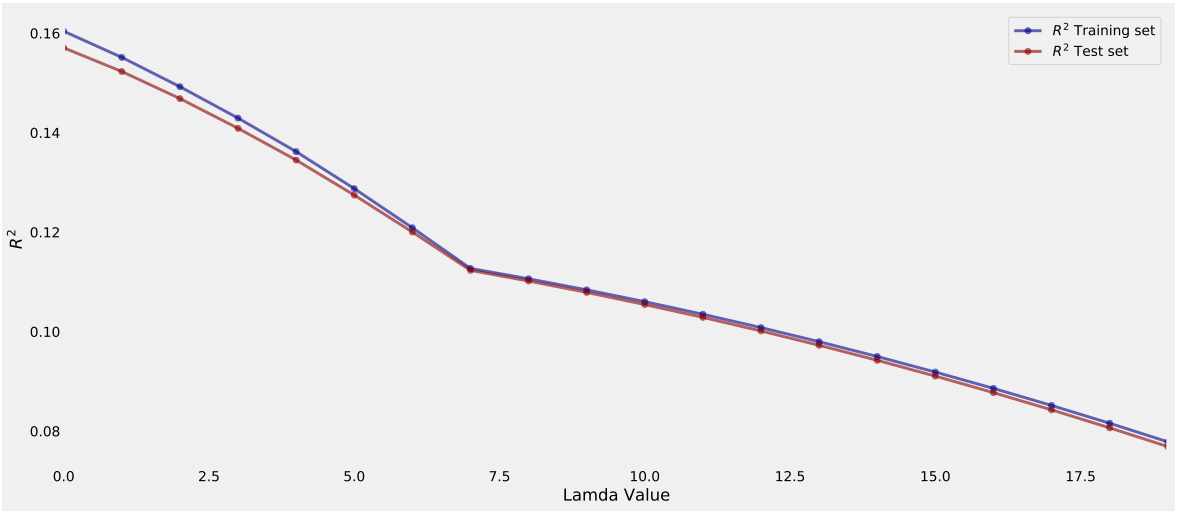

Figure 3.13: Five Fold Cross Validation With Varying Lambda

3.3.3 Ridge Regression

This method is similar to lasso as both try to improve the model by reducing the coefficients' variance. In the case of correlated independent features, lasso regression

may select any of those features at random and remove others. But unlike lasso, ridge regression does not mitigate any coefficient to precisely zero. It provides a mechanism to deal with overfitting where all the features are present, and the model has reduced complexity. The tuning parameter for the technique penalizes the sum of squared coefficients, and hence it belongs to the L2 class of regularization [28]. The cost function to minimize is:

$$J\left(\theta_0, \theta_1\right) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{3.8}$$

Here symbols have the same meaning as defined for linear and lasso regression.Ridge regression estimates a similar value for coefficients of correlating features and achieves suitable results even when a large number of such elements are present in the dataset. The technique works well with ill-conditioned data containing noise and outliers.

Data in a standardized format is one of the requirements for the ridge regression. The difference between the feature value and mean is divided by the standard deviation to get the standardized form. The various implementations of the algorithm support the function internally, and the calculations take place on a standardized scale.

3.4 XGBoost

XGBoost is a scalable, portable, and distributed gradient boosting framework designed for speed and performance. The objective function for the algorithm is a function of functions and cannot be optimized in Euclidean space using conventional optimization techniques [29]. It is hard to explore various base functions and calculate the value of loss function for them because of computation limitations.

The algorithm uses the Taylor series approximation to transform the objective function and get it in the Euclidean domain. The definition of the series at $x = 0$ is:

$$f(x) = f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \cdots = \sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!}x^k \tag{3.9}$$

This method also reduces the computation of calculating the exact loss for various base functions. The algorithm considers the expansion of the series until the second-

order derivative, assuming that the higher derivatives will not add any significant value. Hence objective function reduces to quadratic function in one variable, and traditional convex optimization techniques can solve it.

To solve the issue of a large number of base functions, XGBoost selects the heuristic strategy and builds a tree with the greedy approach by choosing the split giving maximum loss reduction. The algorithm decides from a bunch of split strategies based on a rule-based selection. Several features of XGBoost are listed in the figure 3.14.

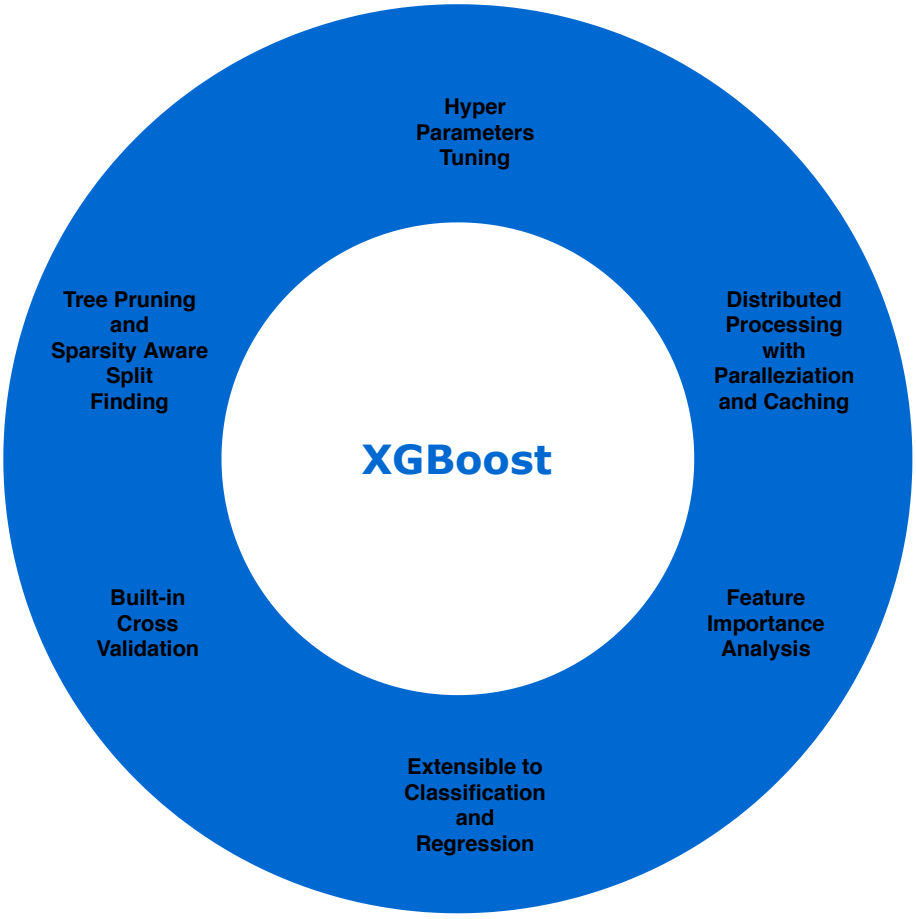

Figure 3.14: Features of XGBoost

Grading boosting algorithms are capable of providing feature importance without additional computation. It is possible because the algorithms construct boosted trees

based on relative importance scores. Every feature receives the score on each decision tree that helps in comparison. Further, the final importances are the average across all the decision trees constructed by the algorithm. The figure 3.15 shows the feature importance for one of our models from the study.

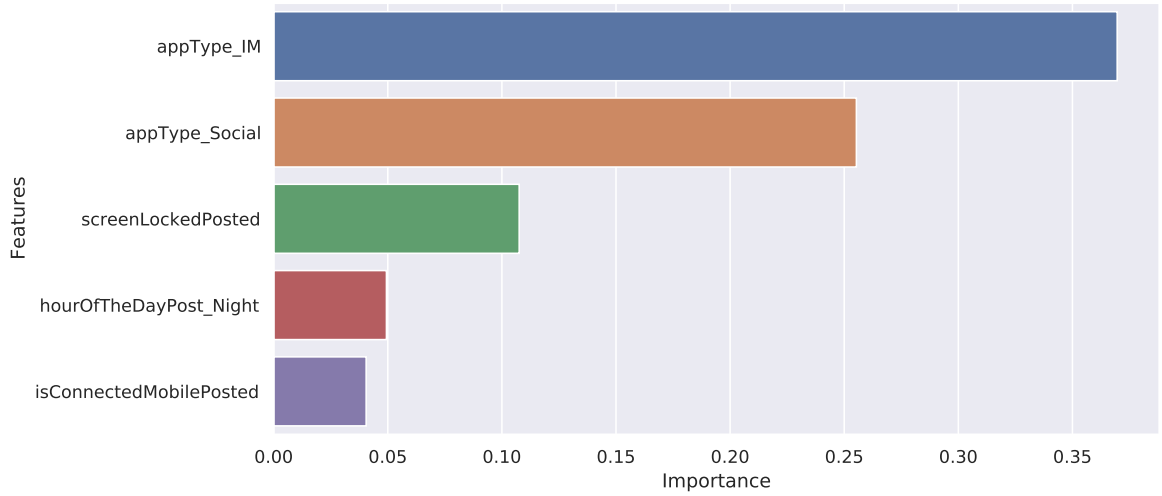

Figure 3.15: Top Five Features Using XGBoost

Building a classifier using XGBoost is simple, but tuning it for the performance is a little complex as it provides many hyperparameters to tune. The algorithm offers the following three types of parameters [30]:

- General Parameters: Help to select the booster models and overall functioning.
- Booster Parameters: Help to tune the booster we choose in general parameters.
- Learning Task Parameters: Help to optimize performance by setting learning scenarios.

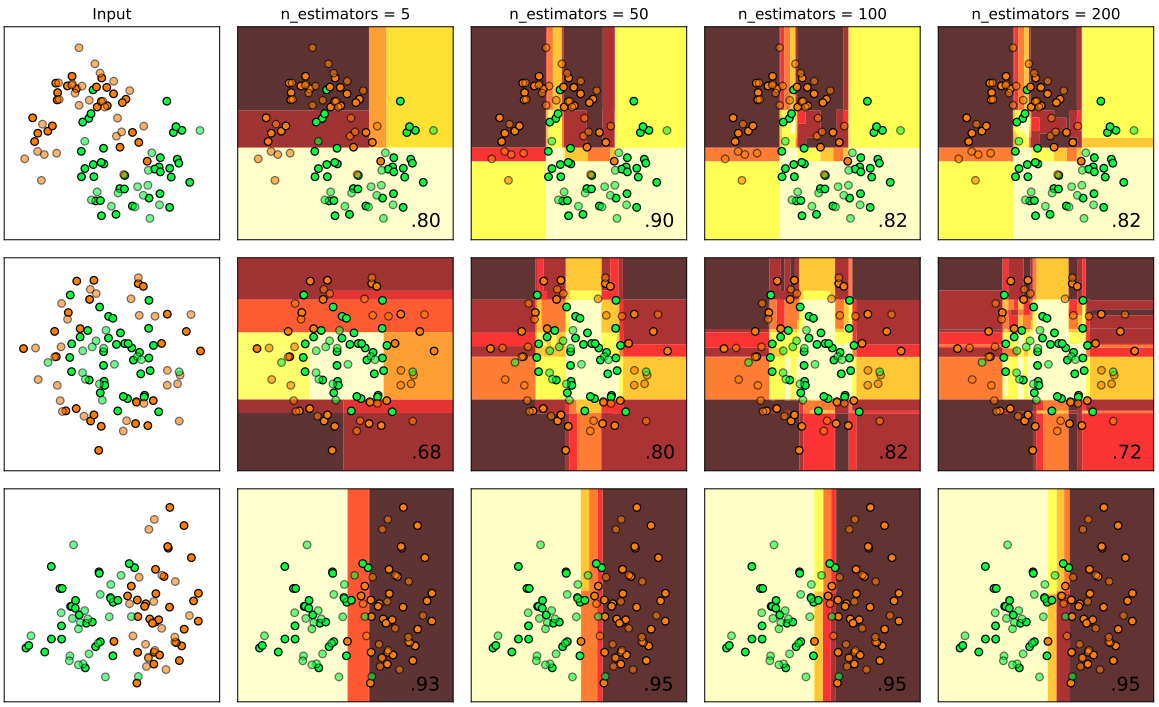

Figure 3.16: Effect of Number of Estimators

We plot the decision boundaries and tweak with three of the most important parameters to understand their effects. Figure 3.16 shows the algorithm's behavior on three different datasets when the n_estimator increases. This parameter sets the number of sequential trees we construct to correct the prior trees.

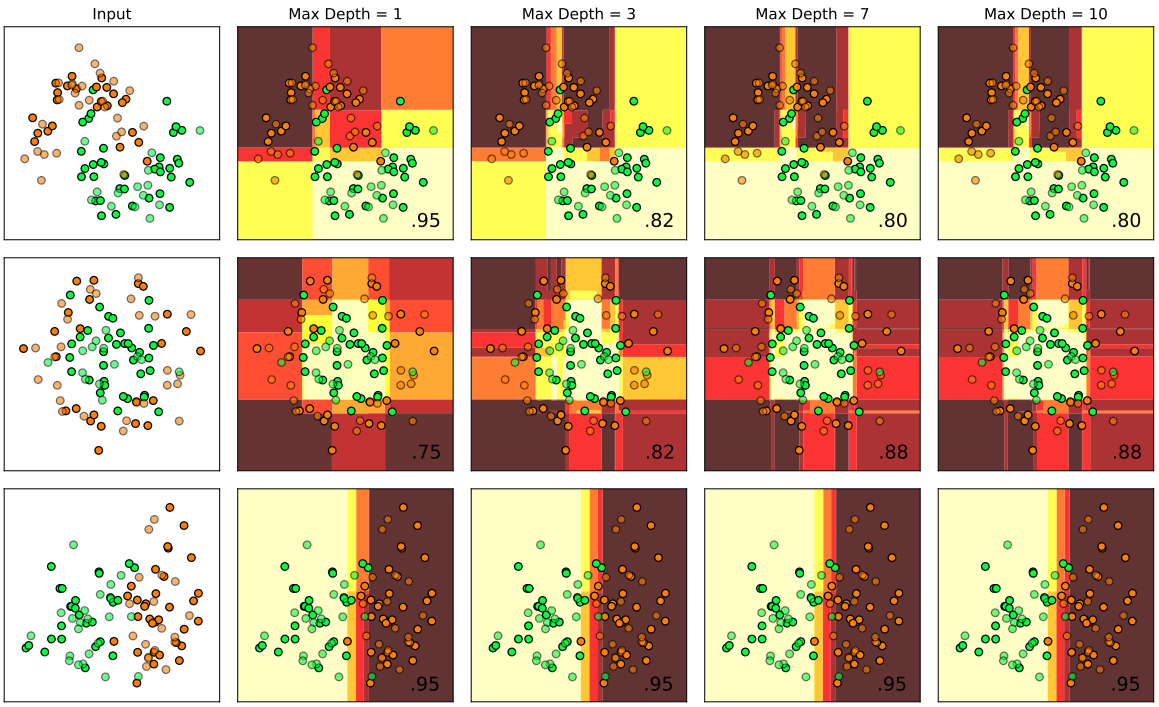

Figure 3.17: Effect of Maximum Depth

We can also set the maximum depth of trees the algorithm constructs to achieve the output. A very large value may lead to overfitting and requires a set number of leaf nodes or minimum child weight to attenuate. Figure 3.17 shows the change in decision boundaries when we increase the depth from one to ten folds.

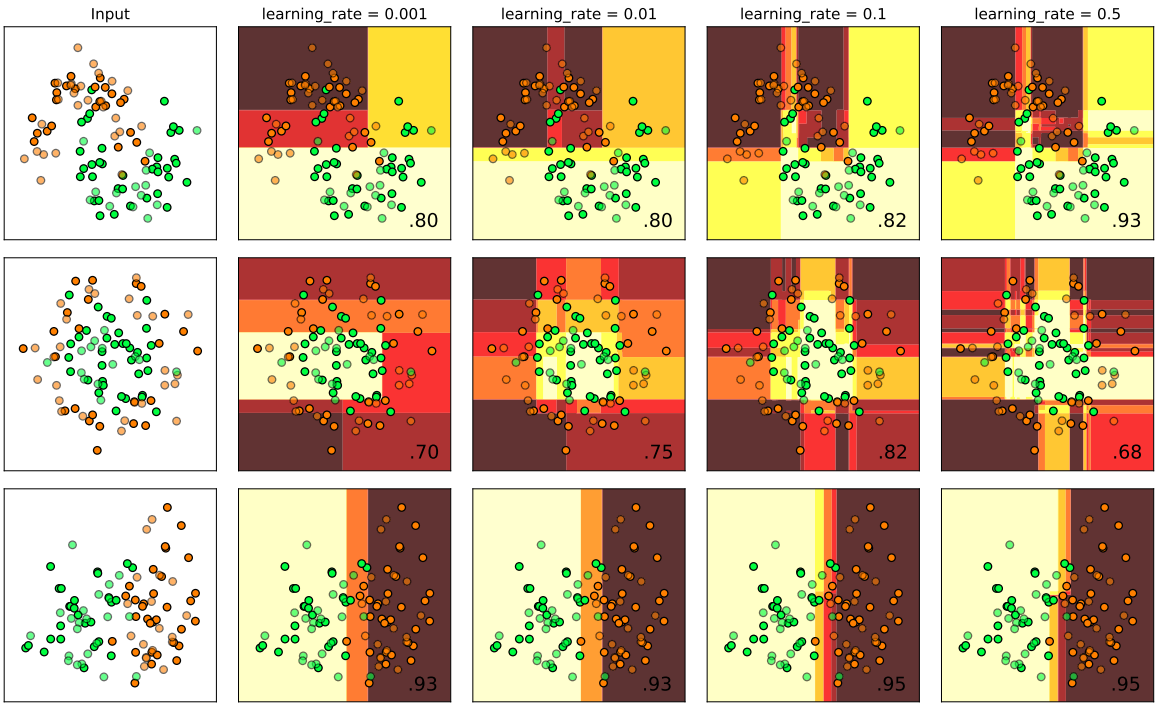

Figure 3.18: Effect of Learning Rates

The learning rate controls the step size for the gradient boosting and helps prevent overfitting. Like other parameters, the value for this also depends on the complexity and size of the dataset. A small amount may take a longer time to train, while a larger may overfit the model. Figure 3.18 shows the classification of two classes on varying the learning rate from 0.001 to 0.5.

It is clear that continuously increasing the value of any parameter does not guarantee an enhanced model but follows the law of diminishing returns. After a certain point, the increment lowers the performance, and suitable conjunction between parameters can give the desired result.

3.5 Evaluation Metrics

While using machine learning algorithms, it is vital to measure the quality of the models. The choice of metric depends on the type of the model, implementation, and hypothesis in the test. A model performing better predictions yield a better metric score and can help tune in the right direction. As our study has classifications as well as regression models, we discuss evaluation metrics for both in the section below.

3.5.1 Classification

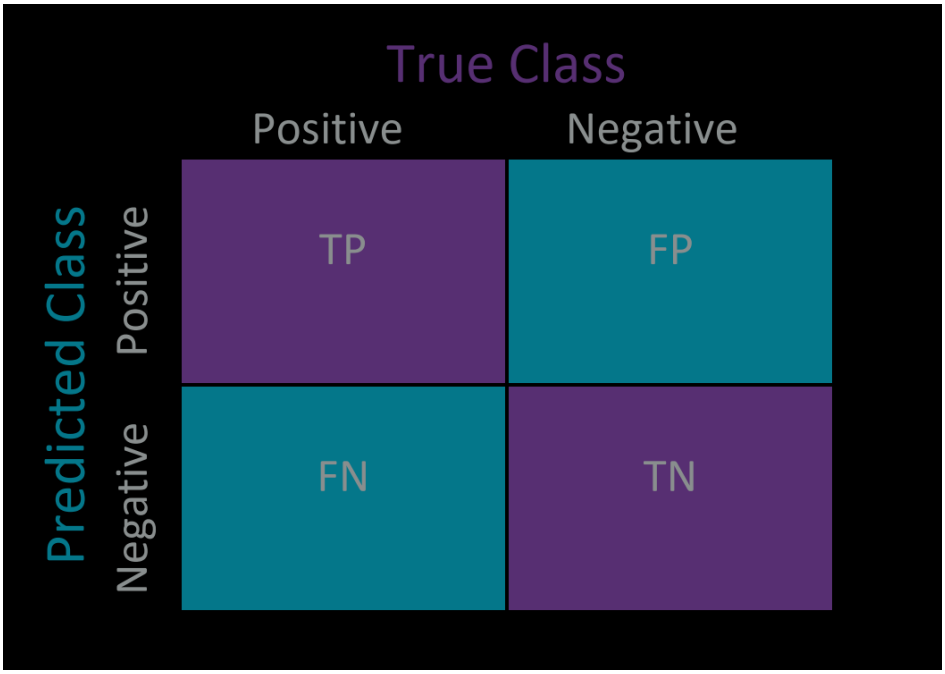

Figure 3.19: Confusion Matrix Components

Confusion Matrix

A Confusion matrix is a square matrix of size N, where N is the number of classes in the dependent variable. It helps us to identify the expected and predicted values count for each category very conveniently. Each cell in the matrix is one of the following value:

True positive (TP): Count increases for the cell when the expected and predicted both holds true.
True negative (TN): Count increases for the cell when the expected and predicted both holds false.
False positive (FP): Count increases for the cell when the expected was false but the predicted value holds true.
False negative (FN): Count increases for the cell when the expected was true but the predicted value holds false.
Figure 3.19 shows the confusion matrix for binary class with the distribution of cells as above.

Precision or Positive Prediction Value (PPV)

It measures the ratio of positive class instances that our model correctly predicts and the total number of predictions. It is useful to use when we have high costs for false positives and low for false negatives. The mathematical formula for the precision is in the equation 4.10.

$$\text{Precision} = \frac{\text{True Positives (TP))}}{\text{True Positives (TP))} + \text{False Positives (FP))}} \tag{3.10}$$

Recall or Sensitivity

The recall is the proportion of actual positive cases which our model correctly predicts. It is useful to use when we have high costs for false negatives, and we want to identify as many positives as possible correctly. Recall alone can not give good metrics of evaluation for a model as it will be the maximum value of 1 if we predict 1 for all instances. The recall fraction is:

$$\text{Recall} = \frac{\text{True Positives (TP))}}{\text{True Positives (TP))} + \text{False Negatives (FN))}} \tag{3.11}$$

F1 Score

To establish a trade-off and get the fittest of recall and precision, we use their har-

monic mean, known as F1-score. We consider the harmonic mean over arithmetic mean as the former penalizes the extreme values more. The equation for F1-score is as follows:

$$F_1 = \left(\frac{\text{Recall}^{-1} + \text{Precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.12}$$

Area Under the Receiver Operating Characteristic Curve (AUC-ROC)

The plot between sensitivity and (1 - specificity) is known as the Receiver Operating Characteristic Curve (ROC). Specificity is the ratio of actual negative cases that our model correctly identifies. Area Under the Curve (AUC) is the quantity that brings the ROC to a comparable single number. The value for AUC lies between 0.5 and 1, where 0.5 signifying the failure of the model, while 1 may represent an overfitting model.

3.5.2 Regression

Root Mean Squared Error (RMSE)

RMSE is the most extensively used evaluation metric for regression models. It highly penalizes the significant deviation from the original value and hence can quantify the large errors. A value closer to 0 is considered an excellent RMSE value. The equation below gives the formula to calculate the RMSE for N number of instances.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (\text{Predicted}_i - \text{Actual}_i)^2}{N}} \tag{3.13}$$

R Squared

Also known as the coefficient of determination, $\hat{\text{R2}}$ helps us compare our model with a random baseline model. It gives the measure of the variability of a dependent variable from its relationship with the independent variable. The value is in the range of 0 and 1 with the worst fit and best fit, respectively. Mathematically it is represented as the

following:

$$R^2 = 1 - \frac{\text{MSE(Model)}}{\text{MSE(Baseline)}} \tag{3.14}$$

In the above equation, MSE(Baseline) gives the Mean Squared Error when all the predictions are the dependent variables' mean value against the actual values.

Chapter 4

Data Preparation and Analysis

4.1 Statistics of Collected Data

Twenty-five people signed up to volunteer for the study and downloaded the notification logger application to their mobile phone. Out of them, 13 people kept the app for 30 days on their phones for whom we present the statistical description.

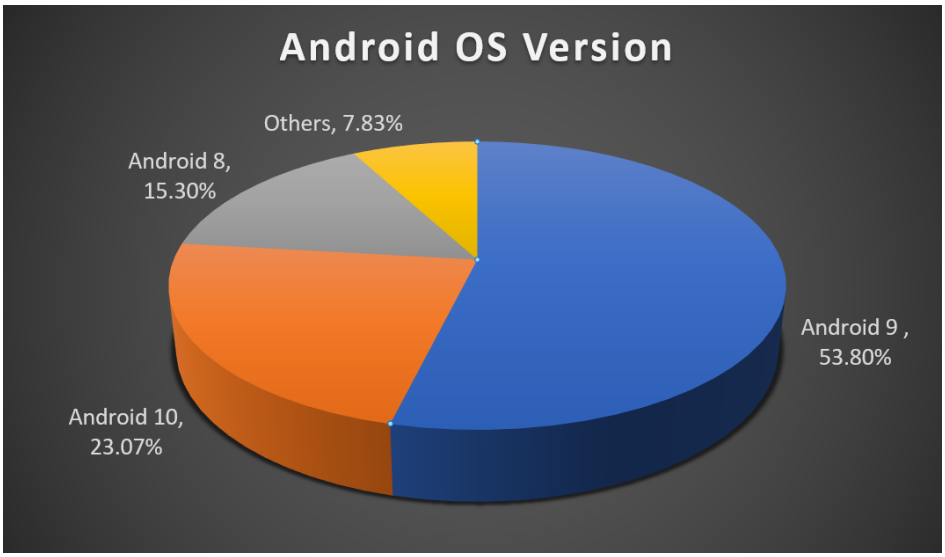

Figure 4.1: Operating System on Volunteers' Phones

The volunteers were from the regions of Europe, North America, and Asia. The

mean age was 28.84, with the minimum age being 21 and a maximum of 56 (standard deviation 10.9). There were nine males and four females and Android Version 9 as the most popular OS on their device. Full OS distribution is in the figure 4.1.

We collected 2,750,387 notification alerts instances. The mean of the alerts over 13 people was 211,568, with a standard deviation of 299,655. The figure 4.2 shows the users receiving the maximum number of notifications during the data collection period.

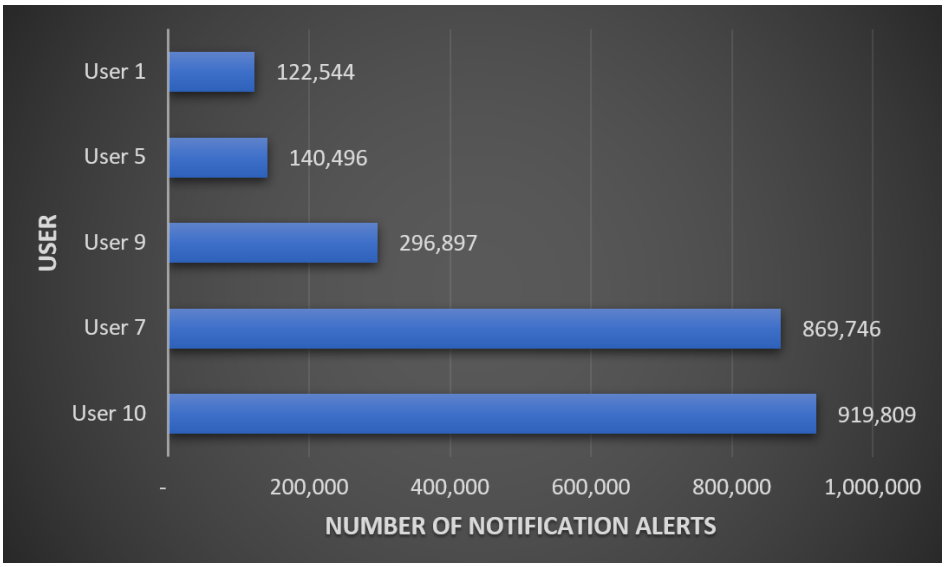

Figure 4.2: Top Five Users With Maximum Notification Alerts

4.2 Data Preparation

4.2.1 Notification Pairing Algorithm

In the preliminary data cleaning, we find that the Android OS generates a notification ID that is not consistent and is recycled to use again. We drop this column from the data. The notification key field looks like the following:

```
0—org.telegram.messenger—-1422345610—null—10186
```

We split the above key string and obtained the application package. Later in the

study, this extraction helped classify the types of different applications and behaviors of the user towards them. We accept a notification for further study if it classifies for one of the following categories:

- Instant Messaging
- Social Media
- Games
- Mail
- Ecommerce
- Media Stream

Forty apps qualify for the study. The list of application packages in each category is available in the appendix.

There are no duplicate observations in the data. A few users did not allow us to capture user activity, and we replaced these null values with the letter 'Q'. The collection of geolocation coordinates was also optional as per the consent form for the study. Nine people opt to allow the geolocations access; we did not see any significant change in their location because of the Covid-19 lockdowns across the globe. It is interesting to note that we have two values for a notification: during arrival and removal. This removal can either be because of one of the following:

1. the user taps on the notification icon and interacts with the next event.
2. the user swipes off and removes the notification without interaction.

After checking for data integrity, we developed an algorithm to identify a set of observations and term it as a notification pair.

```
Input: initialObservations  <- itemset containing observations of either
                                postal or removal type
Output: finalList <- itemset containing notifications paired with matching
                     postal and removal

```

```
begin:
        tracker <- map with the key as application package id and value
                                    as item of type initialObservation

        validPackageMap <- map with the key as one of the selected app type
                           and value as package id
        postedObservation <- object of type initialObservation
        removedObservation <- object of type initialObservation

        foreach item i <- initialObservations  do:
                if package id of i not in validPackageMap:
                        continue
                if type of i is "Posted":
                        if package id of i in tracker:
                                continue
                        else:
                                put i in tracker
                else:
                        if package id of i in tracker:
                                postedObservation <- item from tracker with
                                                        key as package id of i
                                remove package id of i from tracker

                                removedObservation <- i
                                finalObservation <- object generated merging
                                                    postedObservation and
                                                    removedObservation

                                if time difference between postedObservation
                                    and removedObservation >= 1 second:
                                            put finalObservation in finalList

        return finalList

end
```

We do not consider the notifications which are available for less than 1 second in the notification tray. These notifications are usually from services running in the background on the phone and go away without the user interaction. Some of those services are email syncing, application updates, and checking for updates.

The algorithm extracts the pair with the assumption that users are not deliberately engaging with selected apps alerts and ignoring other notifications in the tray. Also, we do not account for the cause of the removal of notifications, and it can be either from the two cases described above. The algorithm concatenates the elements from both the alerts forming the pair. The notification pair item gets the time difference between notification posting and withdrawal as a new feature.

4.2.2 Timestamp Treatment

The time for the notification delivery and removal is in the local timezone of the users. We use the time interval to determine whether the day was a weekday or weekend, where Saturdays and Sundays constitute the weekend. The timestamp also helped us identify the hour of the day and add another feature that was further encoded as a numerical feature. These two features derived from the timestamp gave insights about the usage pattern for users in various demographics.

4.2.3 Outliers Treatment

The distribution of the time difference shows the presence of outliers. These outliers are treatment using the 1.5 Inter Quartile Range (IQR) rule, where we remove instances above 1.5IQR of the third quartile and 1.5IQR below the first quartile. Figure 4.3 shows the distribution of time difference after the treatment.

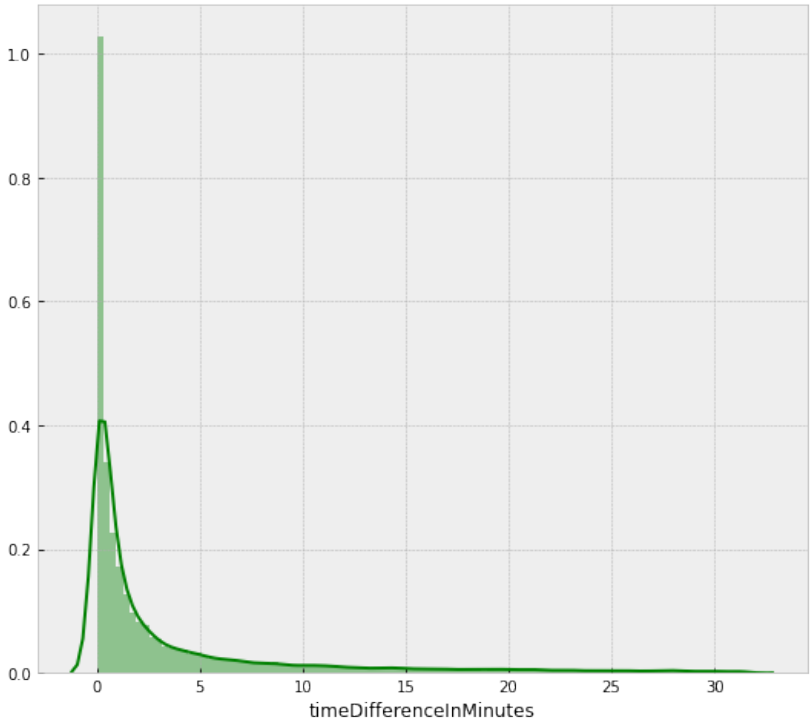

Figure 4.3: Time Difference Distribution After IQR Treatment

4.3 Analysis of Prepared Data

The tight conditions around practical assumptions in the algorithm help to lower the noise and functional limitations. We get real-time notifications instances from the apps into consideration that can help understand the behavior and use patterns. There are a total of 31,059 notifications from the 13 volunteers with properties merged from postal and removal moments. Top 3 users constitute more than 50% of the instances. We plot a few graphs to understand the distribution better.

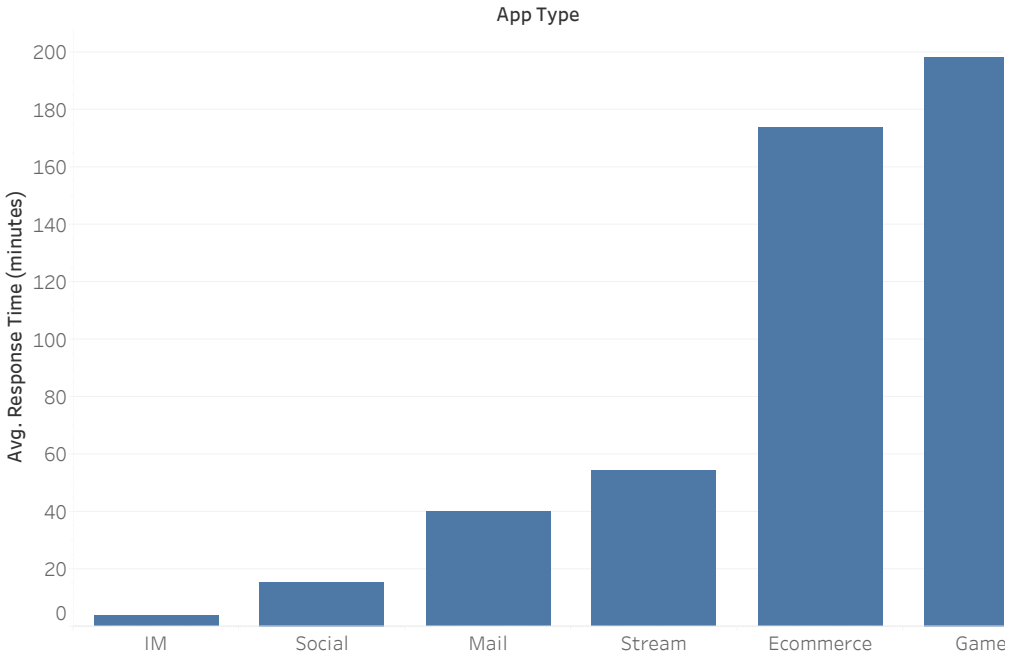

Figure 4.4: Average Response Time for Application Categories

From figure 4.4, it is clear that users respond to instant messaging notifications in the shortest time. Social media applications are the next to capture the attention of the users. People take the maximum time to interact with notifications generated from games as they are mostly promotional and reminders about the update. As shown in figure 4.5, the number of notifications generated from different app types is in the same order except for the stream and mail services. Instant messaging creates the maximum number of notifications while the games are the least. Interestingly, the app type distribution for various users varies; for instance, social media usage for user 1 is significantly more than instant messaging.

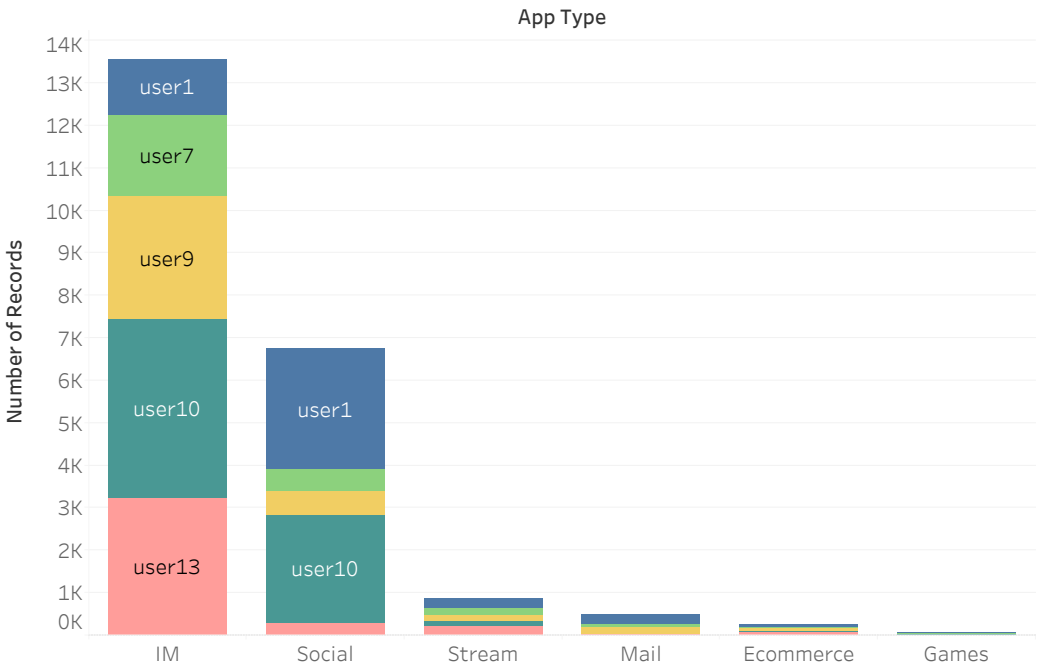

Figure 4.5: Number of Records in Various Application Categories

While analyzing the users' activity when the notifications arrive at their phones, we find a minimal movement. In figure 4.6, we plot the user activity for the users who allowed us to capture their actions. The row with the title "Unknown" resembles when our app was unable to identify the user activity with significant confidence. The impact of staying at home because of the pandemic is distinctly visible. More than 95% of the instances suggest that users were still during the time. Only users 10, 7, and 12 had a little movement on foot, but no one was involved in any other activity.

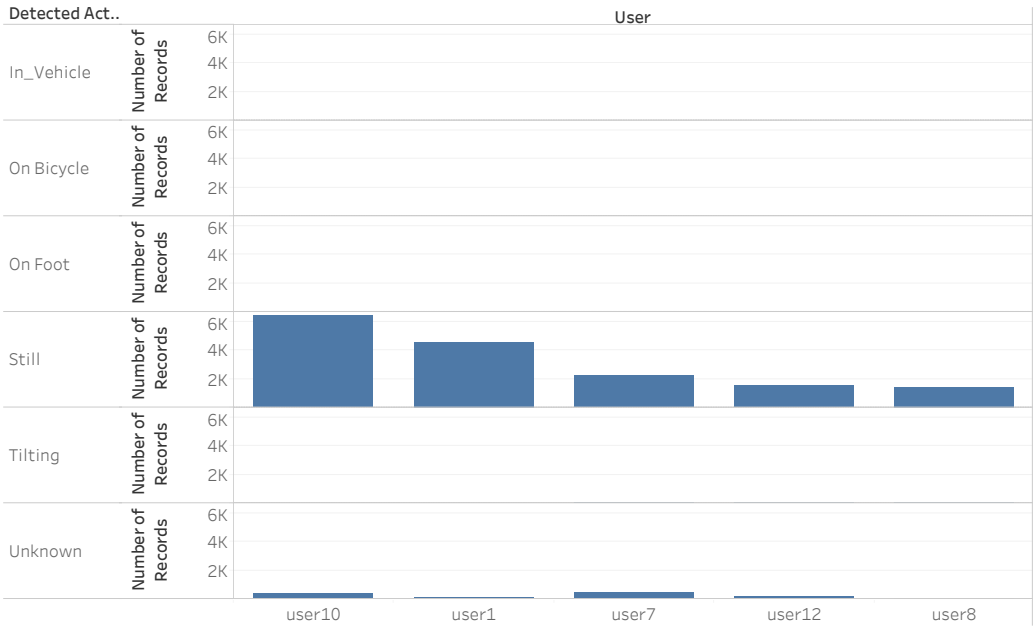

Figure 4.6: Users Current Activity Distribution

While analyzing the users' activity when the notifications arrive at their phones, we find a minimal movement. In figure 4.6, we plot the user activity for the users who allowed us to capture their actions. The row with the title "Unknown" resembles when our app was unable to identify the user activity with significant confidence. The impact of staying at home because of the pandemic is distinctly visible. More than 95% of the instances suggest that users were still during the time. Only users 10, 7, and 12 had a little movement on foot, but no one was involved in any other activity.

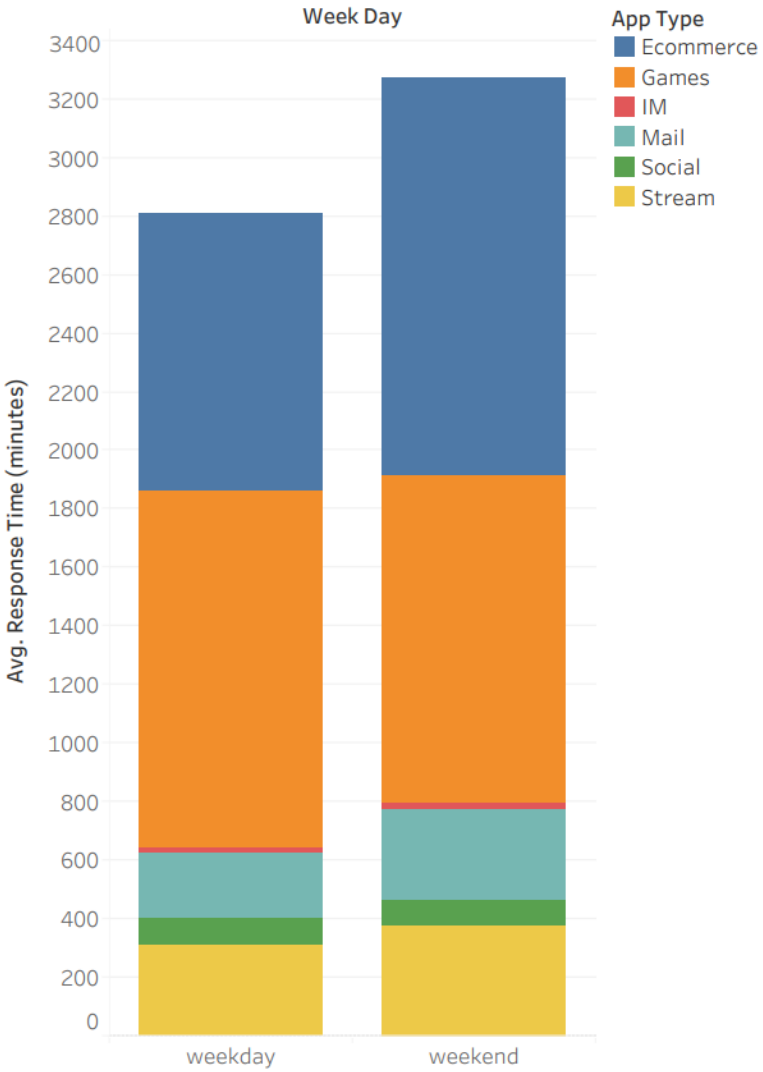

Figure 4.7: Response Time Distribution for Weekdays and Weekends

The response time for various application categories divided for weekend and weekday is in the figure 4.7. Instant messages and social media get quick response irrespective of the day is a weekend or weekday, while users take a longer time to respond to emails on weekends. This suggests that the application type and the weekday can be a crucial feature for the study.

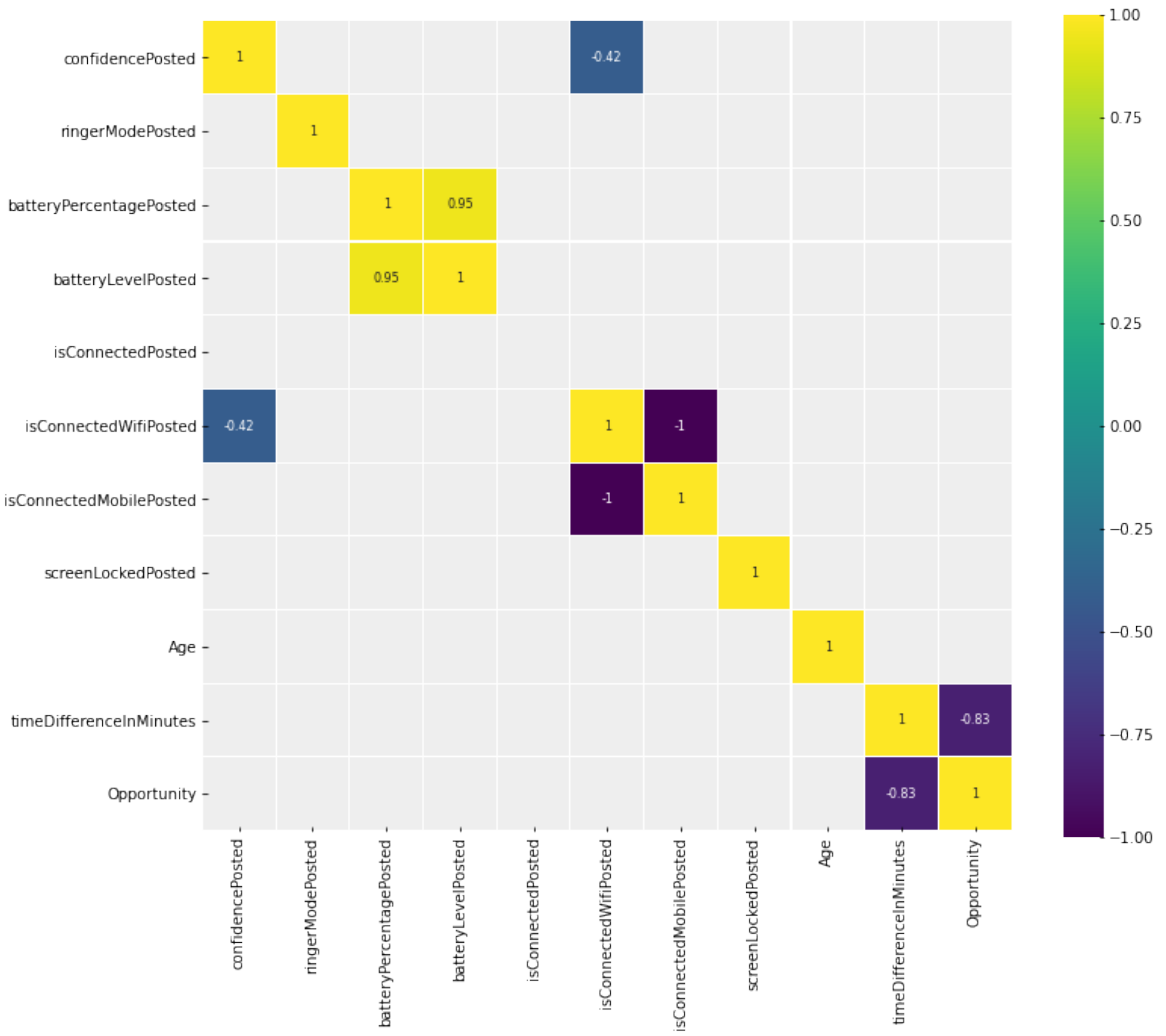

Figure 4.8: Correlation Matrix of Features

The correlation matrix in figure 4.8 gives an interesting relationship between WiFi connection on the user's device and the confidence in predicting the user's activity. The reason for this lies in the implementation of activity recognition API in which the phone uses the WiFi network to enhance the prediction result.

Chapter 5

Model Implementation

The previous chapter described the process of preparing data for our further study. In this chapter, we discuss the three types of models to answer our research question.The dataset holds four categorical features: application type, the hour of the day slot, weekday, and gender. We converted these variables to numerical features using the one-hot encoding technique before building any of the models.

5.1 Binary Class Classification Model

We built a binary class classification model for identifying if a notification delivered at a point of time falls in an opportune moment. The data labeling for a moment to be opportune or not is based on the time difference between posting and removal of a notification. The four algorithms implemented are Logistic Regression, Random Forest Classifier, Support Vector Classifier (SVC), and XGBoost Classifier. We performed ten-fold cross-validation (out-of-sample testing) with Grid Search to achieve the most suitable parameters for each classifier. The kernel type for SVC we chose was Radial Basis Function (RBF), with 0.025 as the regularization parameter for the squared L2 penalty. The probability parameter was set to true to enable probability estimates. Binary logistic was set as the XGBoost Classifier's objective function, and other parameters as the parameter tuning algorithm suggested.

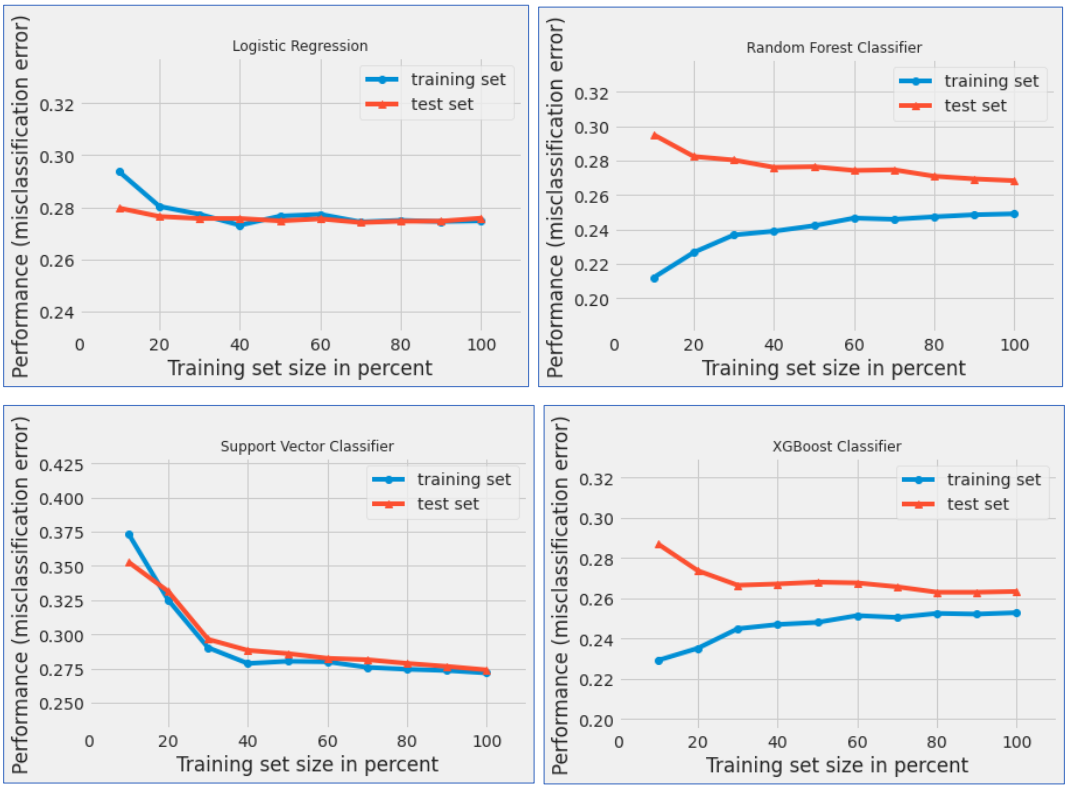

Figure 5.1: Learning Curves for the Binary Class Classification

Figure 5.1 shows the learning curve for the algorithms. A high misclassification error in the test set depicts a bias problem. The direction of bias is obtained from training error. A high training error represents high bias, while a low training error represents little bias. The gap between the two error curves determines the variance, with a broader gap being more variance. Logistic Regression and SVC suffer from underfitting, while Random Forest and XGBoost show a better fit.

5.2 Multi-Class Classification Model

There are four classes in the dependent variable, as described in the previous chapter. We use the same classifiers as we did for the binary class classification but with different parameters to suit the multi-class nature. The Logistic Regression uses the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) algorithm as the solver to optimize the model. Information gain is the criterion function for Random Forest Classifier to measure the quality of a split. The SVC model built with RBF kernel calculates the influence (gamma) and regularization (c) parameter using Grid Search parameter tuning.

Core XGBoost library provides an inbuilt data structure known as Data Matrix (DMatrix) for optimized performance and efficiency. Data frames containing training and testing datasets are converted to DMatrix before building the model. The model uses "softprob" as the objective function, which gives the probability of each class we are predicting. The parameter search provides the best value for the number of estimators, maximum depth, and step size shrinkage. Figure 5.2 shows the feature importance distribution using XGBoost after hyper parameters tuning.

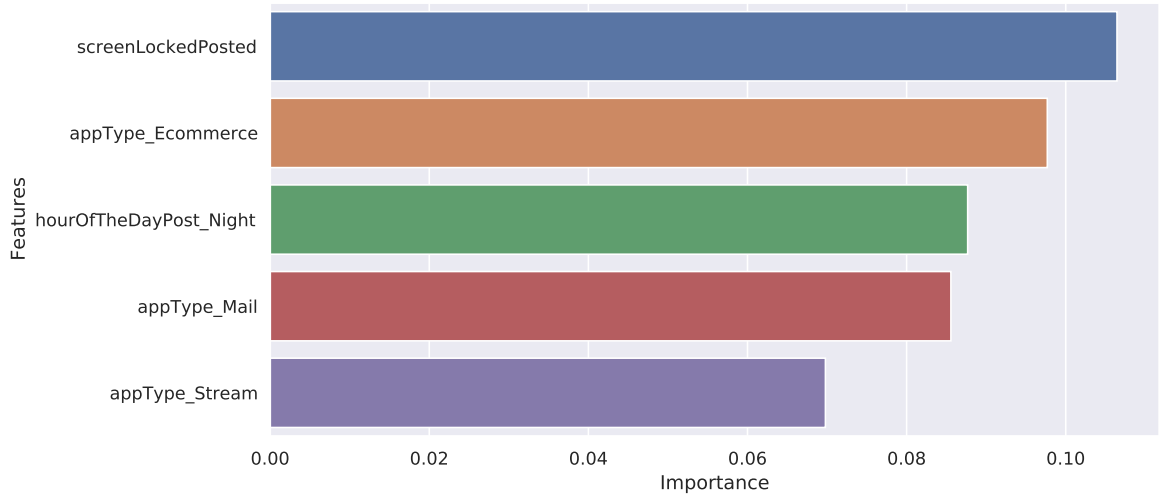

Figure 5.2: Feature Importance for the Multi-Class Classification

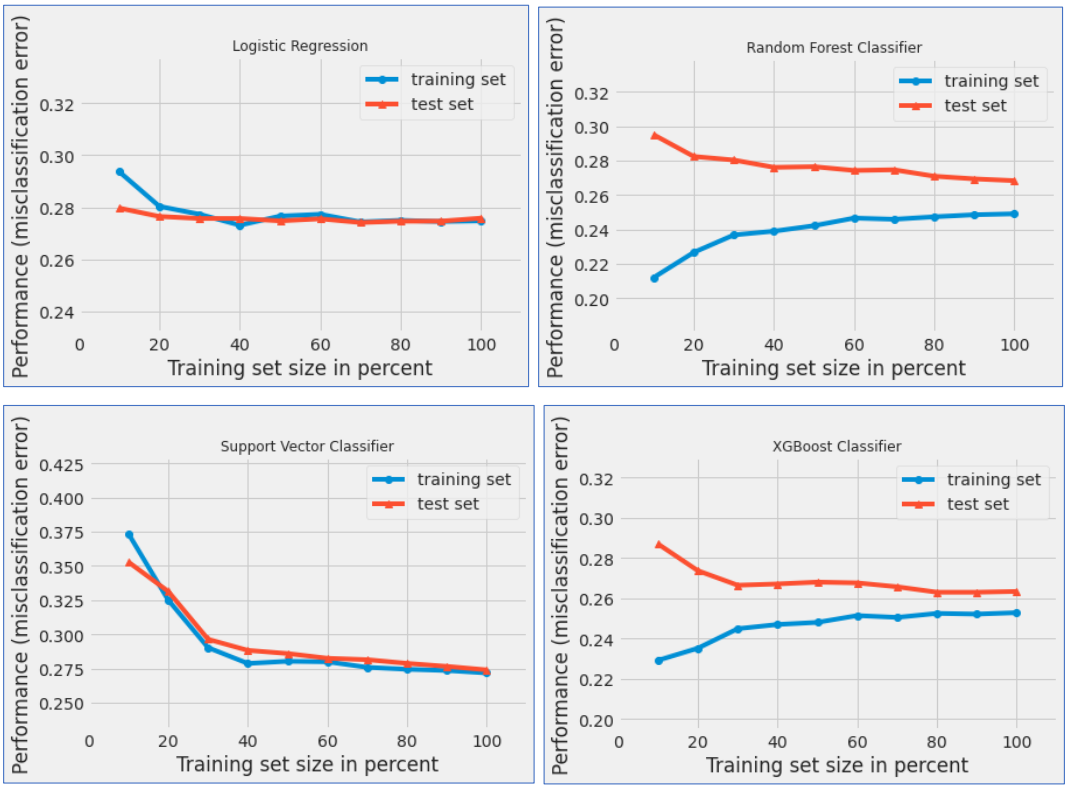

Figure 5.3: Learning Curves for the Multi-Class Classification

Figure 5.3 shows the learning curves for the multi-class classification. Logistic Regression and Support Vector Classifiers fail to converge while ensemble models of Random Forest and XGBoost performs a little better. Though the performance of either model is not very good, adding certain features can enhance the performance. The SVC training time stands out while using a non-linear kernel as the time depends exponentially on the number of features.

5.3 Regression Model

The time difference between postal and removal of the notifications as a feature does not follow a normal distribution and is highly skewed. We apply log transformation to the feature before constructing any regression models. After converting the categorical features to numeric values, we divide the dataset into 75:25 for the training and validation set.

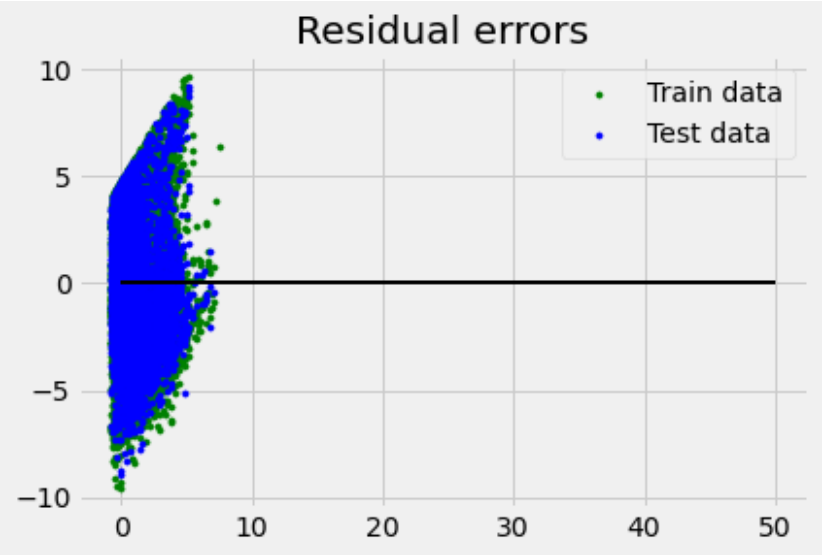

Figure 5.4: Ordinary Least Squares Regression Residual Errors

Ordinary Least Squares regression (OLS) gives the distribution of the residual errors, as shown in figure 5.4. L1 and L2 penalties are introduced in two different models for the enhancement. Grid Search CV is used to find the best parameters for the models. The pipeline is set for linear as well as polynomial fit to obtain the best hyperparameters. The models perform very ordinarily and cannot be considered satisfactory. This is mainly due to limited independent variables. Further, XGBoost regressor also provides similar evaluation results to predict the next opportune moments in minutes.

Chapter 6

Results

The removal of the outliers enhanced the classification and regression models—the number of notifications pairs instances to build the models after the cleaning counts 22,257. We use Dummy Classifier and Regressor from Sklearn as our baseline model that gives predictions based on simple rules.

6.1 Current Moment as Opportune

The identification of the current moment as suitable for notification delivery or not is a binary class classification problem. Logistic Regression and XGBoost Classifiers give similar results for the binary classification. Figure 6.1 shows the confusion matrix for each classifier.

The AUC for classifiers shows a very slight variation and stands at 0.75 as the best obtained by XGBoost as shown in figure 6.2 and 6.3. The table 6.1 gives the metrics comparison for the four classifiers. The baseline classifier uses the stratified strategy and generates the prediction based on the training set's class distribution. All the models perform better than the baseline as seen from the table 6.1.

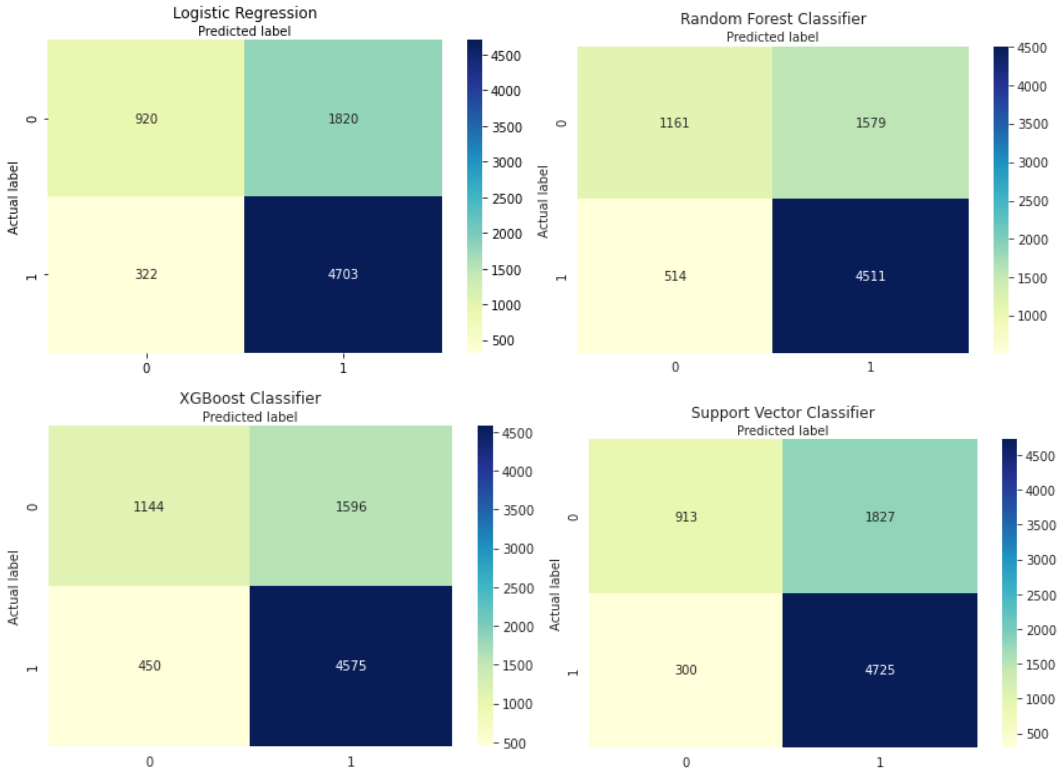

Figure 6.1: Confusion Matrices for Binary Classification

Model	Precision	Recall	F1-Score
Baseline Classifier	0.644	0.647	0.646
Logistic Regression	0.720	0.935	0.814
Random Forest Classifier	0.740	0.897	0.811
Support Vector Classifier	0.721	0.940	0.816
XGBoost Classifier	0.742	0.919	0.818

Table 6.1: Evaluation Metrics for Binary Classification

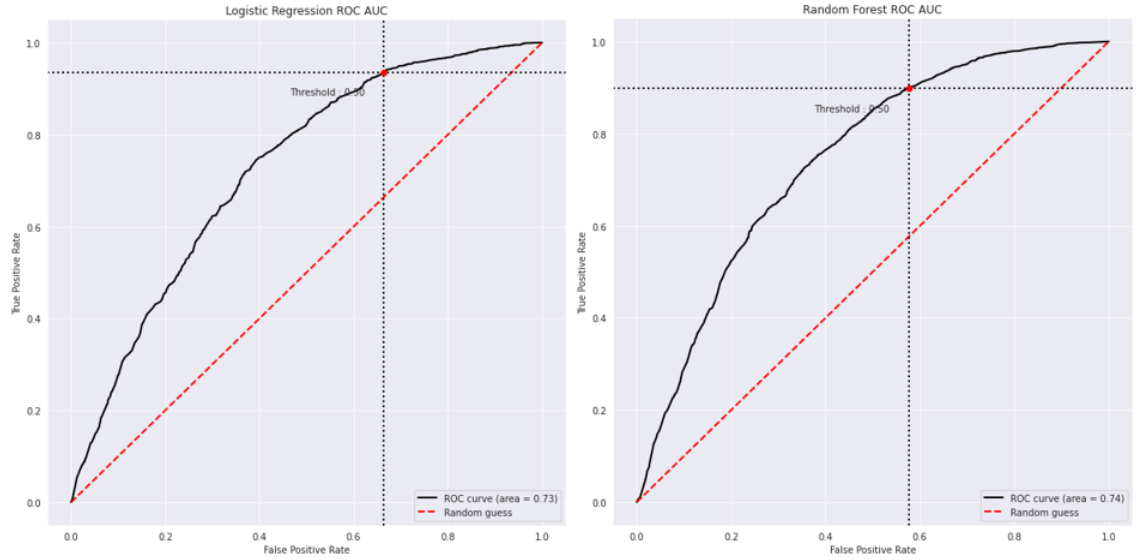

Figure 6.2: ROC Curve for Logistic Regression and Random Forest Classifier

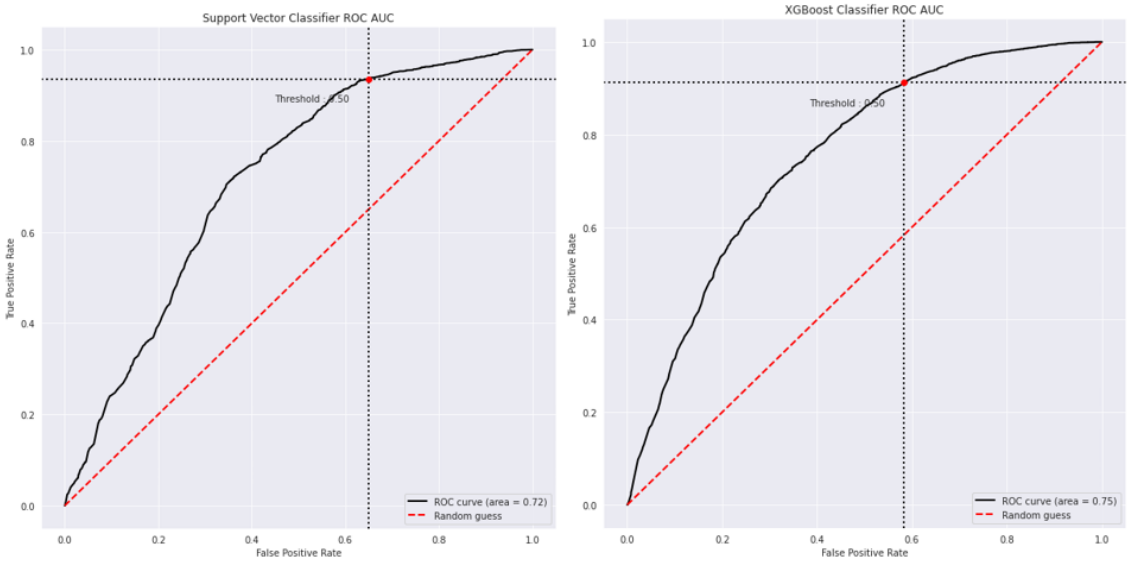

Figure 6.3: ROC Curve for Support Vector Classifier and XGBoost Classifier

6.2 Next Opportune Moment Slot

Multi-class classification to identify the next opportune moment as a slot with various time differences also performed better than the baseline classifier except for the Support Vector Classifier. Figure 6.4 and 6.5 show the confusion matrices distribution for the classifiers. The ensemble models perform slightly better. The classifiers' evaluation metrics are compared based on three average strategies in the table from 6.2 to 6.6. Average micro computes using total true positives, false negatives, and false positives irrespective of the prediction's label. Average macro evaluates for each label and does not account for the label imbalance. In contrast, the weighted average considers the proportion for each class type and reports the weighted average.

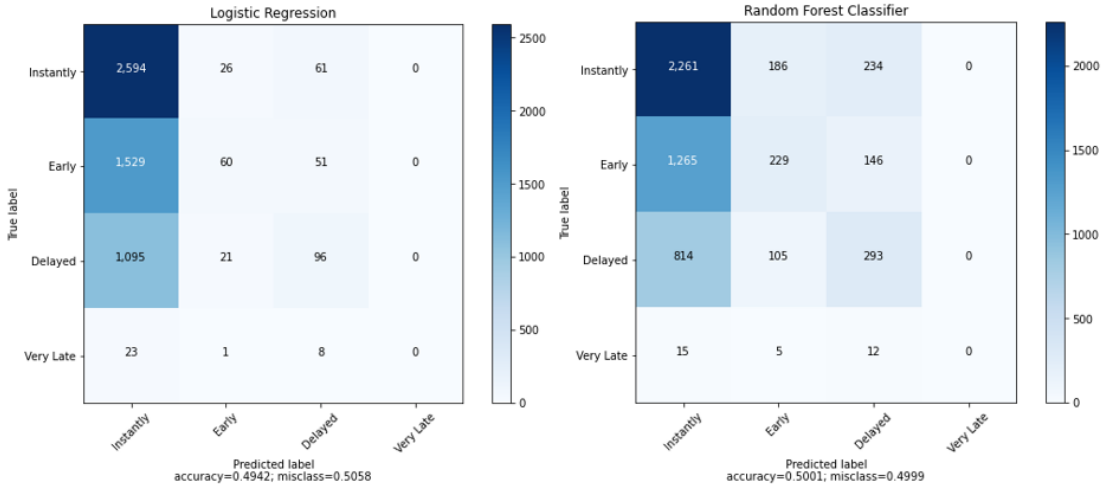

Figure 6.4: Confusion Matrices of Logistic Regression and Random Forest Classifier for Multi-Class Classification

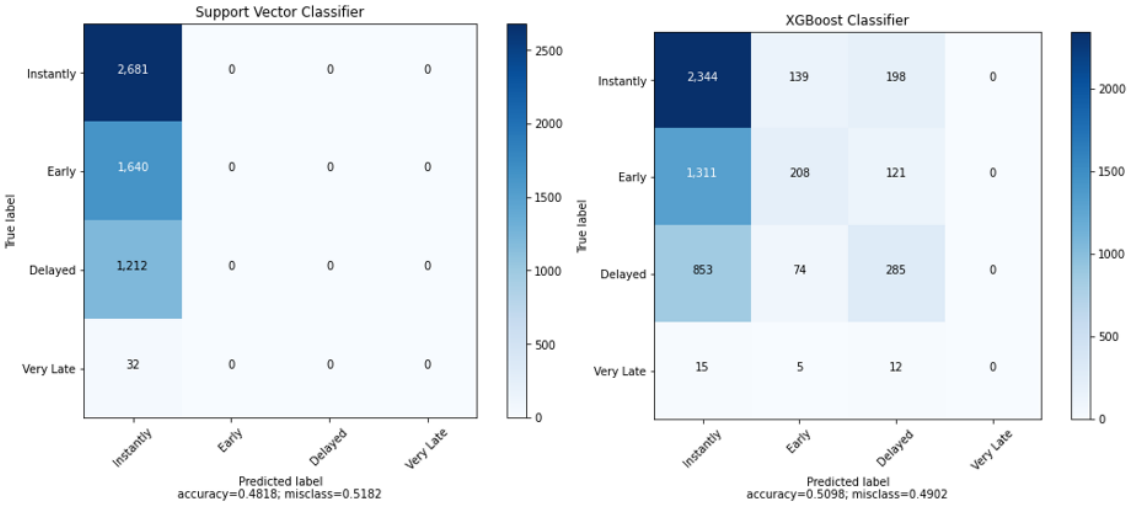

Figure 6.5: Confusion Matrices of Support Vector Classifier and XGBoost Classifier for Multi-Class Classification

Baseline Model			
Average Type	Precision	Recall	F1-Score
weighted	0.369	0.374	0.371
micro	0.374	0.374	0.374
macro	0.251	0.251	0.251

Table 6.2: Baseline Classifier Metrics for Multi-Class Classification

Logistic Regression			
Average Type	Precision	Recall	F1-Score
weighted	0.498	0.494	0.365
micro	0.494	0.494	0.494
macro	0.374	0.271	0.214

Table 6.3: Logistic Regression Metrics for Multi-Class Classification

Random Forest Classifier			
Average Type	Precision	Recall	F1-Score
weighted	0.472	0.500	0.439
micro	0.500	0.500	0.500
macro	0.346	0.306	0.291

Table 6.4: Random Forest Classifier Metrics for Multi-Class Classification

Support Vector Classifier			
Average Type	Precision	Recall	F1-Score
weighted	0.232	0.482	0.313
micro	0.482	0.482	0.482
macro	0.120	0.250	0.163

Table 6.5: Support Vector Classifier Metrics for Multi-Class Classification

XGBoost Classifier			
Average Type	Precision	Recall	F1-Score
weighted	0.494	0.509	0.441
micro	0.509	0.509	0.509
macro	0.367	0.309	0.291

Table 6.6: XGBoost Classifier Metrics for Multi-Class Classification

6.3 Exact Opportune Minute Prediction

The regression models built for predicting the next opportune moment do not converge well for our data. The baseline regressor we have used is from the Sklearn library, with the predictions strategy being the mean of the training dataset. Removal of the outliers and log transformation decreased the RMSE value by 4%, but none of the models could perform significantly well when compared to the baseline model. Figure 6.6 shows the normalized relative prediction pattern for 25 instances. Table 6.7 compares the performance of the regressor models.

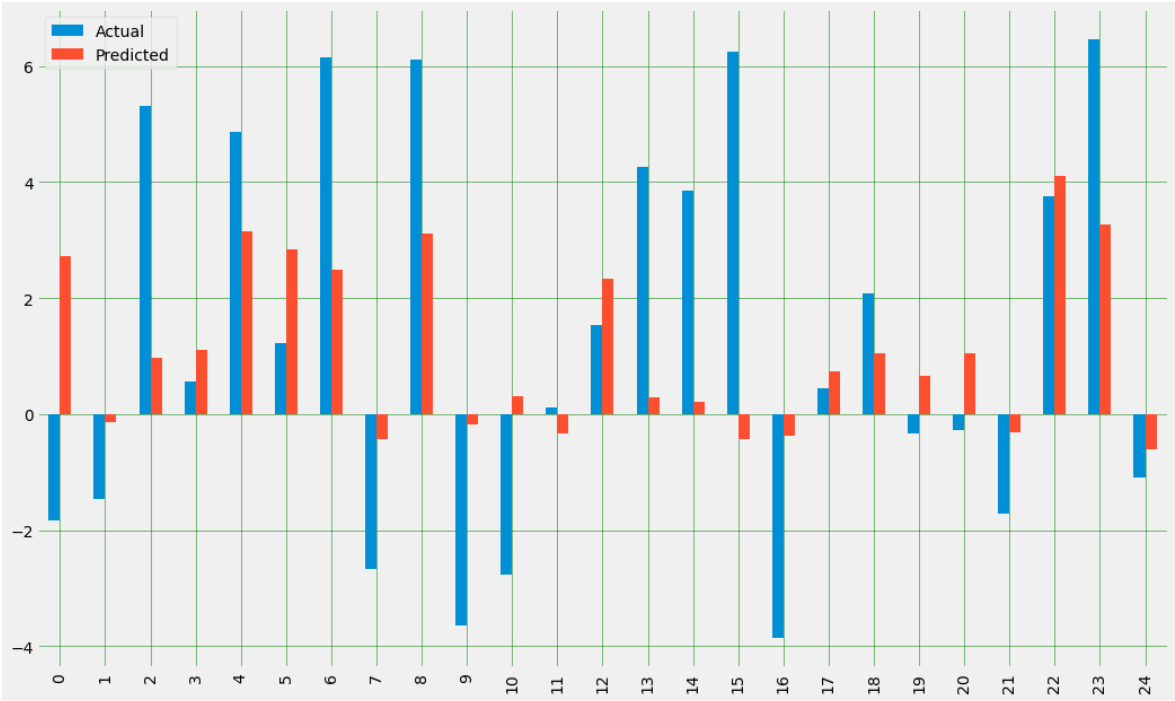

Figure 6.6: Actual Versus Predicted Value Comparison for 25 Instances

Model	RMSE	R2
Baseline Regressor	445.6	0.249
Linear Regression	437.8	0.245
Lasso Regression	445.4	0.247
Ridge Regression	440.2	0.242
XGBoost Classifier	432.8	0.252

Table 6.7: Evaluation Metrics for Regression

Chapter 7

Conclusion

This study developed a data collection framework for intelligent notification systems as per the ethical guidelines. The data collection application ran for around four weeks on thirteen volunteers' devices to collect data points and recorded 2,750,387 notifications instances. The data capture and anonymization followed the GDPR norms. The study also presents an algorithm that reduces noise and overcomes practical limitations to generate notifications pairs for building machine learning models.

Further, machine learning models were implemented to predict the opportune moments for a notification alert delivery on a mobile device. The binary classification gave good results in predicting whether a moment is opportune or not by performing 26% better than the baseline classifier. Prediction of the next opportune moment in multi-class classification gave slightly better (4.5%) performance than the baseline multi-class classifier. The exact time prediction in minutes for the next opportune moment could not be obtained satisfactorily. The reason for the model's average performance is the limited feature range we were able to include in the study. Geo locations and current user activity became irrelevant for the study because of the pandemic and global lockdown.

Chapter 8

Future Work

The models implemented may perform exceptionally well if geolocations and current user activity are available. The developed data collection framework, as well as the models, can be tested post-pandemic caused by COVID-19 to understand these features' importance without any modification.

The notifications can either be human-generated or system generated. For instance, considering a social media application, users can get a notification alert when their friend send a message, or even when a post is trending in their circle. A friend's notification is the case of a human-generated alert, while the latter is the case of system generated notification from the same application. The identification between users and system-generated notifications and their impact on the model's performance can be explored. The cause of removal of the notification (swipe off or opening the app) can serve as a characteristic to compute additional complex variables like the engagement rate. However, this would require a lot more volunteers to gather more numerous data points.

Mobile phones have limited resources, and the operation of models for intelligent notification systems can be challenging. Given the security and privacy concerns associated with the model's data points, the deployment over the cloud may not be the best methodology. Further study can reveal a better mechanism toward the implementation for practical use.

Bibliography

[1] "Deloitte release latest Mobile Consumer stats." https://www2.deloitte.com/ie/en/pages/technology-media-and-telecommunications/articles/mobile-consumer-stats.html, 2020. Accessed: 2020-09-04.

[2] "Notification, Disruption, and Memory: Effects of Messaging Interruptions on Memory and Performance." https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/Interact2001Messaging.pdf, 2001. Accessed: 2020-09-04.

[3] R. K. Garrett and J. N. Danziger, "Im interruption management instant messaging and disruption in the workplace," *Journal of Computer-Mediated Communication*, vol. 13, no. 1, p. 23–42, 2007.

[4] "Firebase Cloud Messaging." https://firebase.google.com/docs/cloud-messaging, 2020. Accessed: 2020-09-04.

[5] B. Efron and R. Tibshirani, "An introduction to the bootstrap," 1994.

[6] S. T. Iqbal and E. Horvitz, "Notifications and awareness," *Proceedings of the 2010 ACM conference on Computer supported cooperative work - CSCW 10*, 2010.

[7] A. Mehrotra, M. Musolesi, R. Hendley, and V. Pejovic, "Designing content-driven intelligent notification mechanisms for mobile applications," *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp 15*, 2015.

[8] A. Mehrotra, V. Pejovic, J. Vermeulen, R. Hendley, and M. Musolesi, "My phone and me," *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016.

[9] A. Mehrotra, R. Hendley, and M. Musolesi, "Interpretable machine learning for mobile notification management," *GetMobile: Mobile Computing and Communications*, vol. 21, no. 2, p. 35–38, 2017.

[10] A. P. Felt, S. Egelman, and D. Wagner, "Ive got 99 problems, but vibration aint one," *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices - SPSM 12*, 2012.

[11] M. Pielot, K. Church, and R. D. Oliveira, "An in-situ study of mobile phone notifications," *Proceedings of the 16th international conference on Human-computer interaction with mobile devices services - MobileHCI 14*, 2014.

[12] J. E. Fischer, N. Yee, V. Bellotti, N. Good, S. Benford, and C. Greenhalgh, "Effects of content and time of delivery on receptivity to mobile interruptions," *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services - MobileHCI 10*, 2010.

[13] H. Oh, L. Jalali, and R. Jain, "An intelligent notification system using context from real-time personal activity monitoring," *2015 IEEE International Conference on Multimedia and Expo (ICME)*, 2015.

[14] M. Pielot, B. Cardoso, K. Katevas, J. Serrà, A. Matic, and N. Oliver, "Beyond interruptibility," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, p. 1–25, 2017.

[15] J. E. Fischer, C. Greenhalgh, and S. Benford, "Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications," *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI 11*, 2011.

[16] V. Pejovic and M. Musolesi, "Interruptme," *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp 14 Adjunct*, 2014.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine

learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] A. Mehrotra, R. Hendley, and M. Musolesi, "Notifymehere," *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*, 2019.

[19] J. Z. Gao and A. Ji, "Smartmobile-ad: An intelligent mobile advertising system," *2008 The 3rd International Conference on Grid and Pervasive Computing - Workshops*, 2008.

[20] K. Doherty, J. Marcano-Belisario, M. Cohn, N. Mastellos, C. Morrison, J. Car, and G. Doherty, "Engagement with mental health screening on mobile devices," *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI 19*, 2019.

[21] K. Doherty, A. Balaskas, and G. Doherty, "The Design of Ecological Momentary Assessment Technologies," *Interacting with Computers*, 08 2020. iwaa019.

[22] "Gartner Says Worldwide Sales of Smartphones Grew 9 Percent in First Quarter of 2017." `https://www.gartner.com/en/newsroom/press-releases/2017-05-23-gartner-says-worldwide-sales-of-smartphones-grew-9-percent-in-fir` 2017. Accessed: 2020-09-04.

[23] S. S. Kumbhar, Y. Lee, and J. Yang, "Hybrid encryption for securing shared-preferences of android applications," *2018 1st International Conference on Data Intelligence and Security (ICDIS)*, 2018.

[24] "Activity Recognition API." `https://developers.google.com/location-context/activity-recognition`, 2019. Accessed: 2020-09-04.

[25] B. G. Tabachnick and L. S. Fidell, *Using multivariate statistics*. Pearson, 2013.

[26] T. K. Ho, "Random decision forests," *Proceedings of 3rd International Conference on Document Analysis and Recognition*.

[27] R. Tibshirani, "Regression shrinkage and selection via the lasso: a retrospective," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 73, no. 3, p. 273–282, 2011.

[28] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 42, no. 1, p. 80–86, 2000.

[29] T. Chen and C. Guestrin, "Xgboost," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[30] "XGBoost Documentation - xgboost 1.3.0-SNAPSHOT documentation." `https://xgboost.readthedocs.io/en/latest/`, 2020. Accessed: 2020-09-04.

Appendix

Additional Tables

Package Name	Category
in.amazon.mShop.android.shopping	Ecommerce
net.one97.paytm	Ecommerce
com.flipkart.android	Ecommerce
com.myntra.android	Ecommerce
com.fungames.sniper3d	Games
com.ludo.king	Games
com.pieyel.scrabble	Games
com.tencent.ig	Games
com.quizup.core	Games
us.zoom.videomeetings	IM
org.telegram.messenger	IM
com.facebook.orca	IM
com.whatsapp	IM
com.android.messaging	IM
com.google.android.talk	IM
com.google.android.gm	Mail
com.google.android.gms	Mail
com.instagram.android	Social
com.tinder	Social
com.facebook.katana	Social

com.twitter.android	Social
com.pinterest	Social
com.reddit.frontpage	Social
com.linkedin.android	Social
com.zhiliaoapp.musically	Social
com.discord	Social
com.jio.media.jiobeats	Stream
com.ace.tv	Stream
in.startv.hotstar	Stream
com.google.android.youtube	Stream
com.spotify.music	Stream
com.netflix.mediaclient	Stream
com.amazon.avod.thirdpartyclient	Stream
com.google.android.music	Stream
com.amazon.mp3	Stream
tv.accedo.airtel.wynk	Stream
com.oppo.music	Stream
com.phonepe.app	Ecommerce
com.mobstac.thehindu	News
tv.accedo.airtel.wynk	Stream

Table 1: List of Applications Selected For Study