



Daffodil
International
University

**Measuring the Impact of Web Performance Issues on Visual Progression
Based User Experience**

Supervised by

Md. Anwar Hossen

Lecturer (Senior Scale)

Department of Software Engineering

Daffodil International University

Submitted by

Shunjid Rahman Showrov

ID: 171-35-1862

Department of Software Engineering

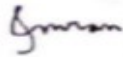
Daffodil International University

The Thesis report has been submitted in fulfillment of the requirements for the Degree of
Bachelor of Science in Software Engineering.

APPROVAL

This thesis is being titled as “**Measuring the Impact of Web Performance Issues on Visual Progression Based User Experience**” submitted by **Shunjid Rahman Showrov, 171-35-1862** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and for approval as per it’s inner styles and contents inside.

BOARD OF EXAMINERS



Dr. Imran Mahmud
Associate Professor and Head
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Chairman



K. M. Imtiaz-Ud-Din
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 1



Md Fahad Bin Zamal
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2




Professor Dr. Md. Nasim Akhtar
Professor
Department of Computer Science and Engineering
Dhaka University of Engineering and Technology, Gazipur

External Examiner

DECLARATION

It's been declared that this thesis including all the scientific experimental works has been completed by me under the supervision of **Md. Anwar Hossen, Lecturer (Senior Scale), Department of Software Engineering of Daffodil International University**. I also declare that neither this thesis nor any part of this whole scientific experiment has been submitted elsewhere for award of any degree.


02.02.2021

Name: Shunjid Rahman Showrov

Student ID: 171 – 35 – 1862

Batch: 22

Department of Software Engineering,
Faculty of Science and Information Technology,
Daffodil International University.

Certified by:


30.1.21

Md. Anwar Hossen
Lecturer (Senior Scale),
Department of Software Engineering,
Faculty of Science and Information Technology,
Daffodil International University.

ACKNOWLEDGMENT

I am grateful to the Almighty Allah (SWT) for allowing me to complete this Bachelor of Science study with this project based research. I am thankful to my Mother who has always been a support for me throughout my life. She is the person who always let me have faith and belief that I can accomplish something. I want to thank my Father who has always been supporting me. I want to thank my younger sister who always inspires me for doing something creative.

I would also love to show my respect and gratitude to my Supervisor, **Mr. Md. Anwar Hossen**, supporting me in completing this research and encouraging me in development and implementation of this work.

Besides, I would like to thank **Dr. Imran Mahmud**, Associate Professor & Head In-Charge of the Department of Software Engineering, Daffodil International University for motivating us for quality research. I also want to thank my teacher who has been continuously supporting us throughout this undergrad. Last but not the least, I want to thank all the stuffs of the Department of Software Engineering and Daffodil International University for continuously working hard to provide us the best facilities in classrooms and laboratories.

TABLE OF CONTENTS

ACKNOWLEDGMENT	III
LIST OF TABLES	VII
LIST OF FIGURES	VIII
ABSTRACT	X
CHAPTER 1: INTRODUCTION	1
1.1 Background.....	1
1.2 Motivation of the Research.....	1
1.3 Problem Statement.....	2
1.4 Research Question.....	2
1.5 Research Objectives.....	2
1.6 Research Scope.....	3
1.7 Thesis Organization.....	3
CHAPTER 2: LITERATURE REVIEW	4
2.1 Case Study on Web Performance Optimization.....	4
2.2 Case Study on Quality of User Experience.....	5
2.3 Trends and Patterns Among Case Studies.....	7
2.4 Addressable Improvements.....	7
CHAPTER 3: RESEARCH METHODOLOGY	9
3.1 Data Requirement.....	10
3.2 Data Collection.....	11
3.3 Data Preparation.....	14
3.4 Exploratory Data Analysis.....	15
3.5 Feature Selection.....	26

3.6 Algorithms Used.....	28
3.6.1 F-Test Regression.....	28
3.6.2 Mutual Information Regression.....	30
3.6.3 Multiple Linear Regression.....	30
3.6.4 Tree Based Regression.....	32
3.7 Re-Sampling.....	34
3.7.1 Re-Sampling by Train-Test-Split.....	34
3.7.2 K-Fold Cross Validation.....	35
3.8 Evaluation and Generalization.....	36
3.8.1 Mean Absolute Error.....	36
3.8.2 Root Mean Squared Error.....	36
3.8.3 R-Squared.....	37
3.8.4 Generalization.....	37
3.9 API Development.....	38
3.10 Deployment.....	39
CHAPTER 4: RESULTS AND DISCUSSION.....	40
4.1 Univariate Feature Ranking.....	40
4.2 Linear Regression Train-Test-Split Re-sampling.....	41
4.3 Linear Regression K-Fold Cross Validation.....	43
4.4 Wrapper Method Significance.....	45
4.5 Decision Tree Regression Re-sampling.....	46
4.5.1 Generalization.....	46
4.5.2 8-Fold Cross Validation.....	47
4.6 Random Forest Regression Re-Sampling.....	48

4.6.1 Best fold estimation.....	49
4.6.2 Best Total Estimators.....	49
4.7 Decision Making.....	52
4.8 Business Benefits of Conversion to PWA.....	54
4.9 GitHub Actions Preview.....	55
CHAPTER 5: CONCLUSION AND RECOMMENDATION.....	56
5.1 Findings.....	56
5.2 Contributions.....	56
5.3 Recommendation on future works.....	57
REFERENCES.....	58
APPENDIX – A.....	60
List of Abbreviation.....	60
APPENDIX – B.....	61
Plagiarism Report.....	61

LIST OF TABLES

Table 2.1: Addressable Improvements Figured from Case Studies.....	8
Table 3.1: Categories of Sites for Which Performance Reports Are Collected.....	12
Table 3.2: Data Attributes and Details.....	13
Table 3.3: Mean of features and target.....	15
Table 3.4: Median of features and target.....	16
Table 4.1: Univariate Feature Significance Ranking.....	40
Table 4.2: Best Predictor Feature Set for Linear Regression with Train-Test-Split....	41
Table 4.3: Linear Regression 3-Fold Cross Validation Summary.....	43
Table 4.4: Linear Regression 5-Fold Cross Validation Summary.....	44
Table 4.5: Linear Regression 5-Fold Cross Validation Summary.....	44
Table 4.6: Lowest Generalization Error In Decision Tree Regression.....	46
Table 4.7: Best R-Squared and RMSE for RFE and UFS in Decision Tree.....	47
Table 4.8: Final Comparison Among Best Candidates.....	52

LIST OF FIGURES

Figure 2.1: Measurement of Web User Experience Trends Over Time.....	7
Figure 3.1: Proposed Research and Development Methodology.....	9
Figure 3.2: Data Collection Process from Page Speed Insights REST API.....	11
Figure 3.3: Data Preparation Steps.....	14
Figure 3.4: Variability of Interactivity and User Experience.....	18
Figure 3.5: Pearson Correlation Co-efficient Among Attributes.....	20
Figure 3.6: Layout Shifting based Correlation (Interactive & UX Index).....	21
Figure 3.7: Largest Content based Correlation (Interactive & UX Index).....	22
Figure 3.8: Distribution of Visual Progression.....	23
Figure 3.9: Distribution of Visual Progression Based on Layout Shift.....	24
Figure 3.10: Distribution of Visual Progression Based on Largest Content Paint.....	25
Figure 3.11: Test Driven Feature Selection.....	27
Figure 3.12: Linear Regression.....	30
Figure 3.13: Train Test Split Re-sampling.....	34
Figure 3.14: K-Fold Cross Validation.....	35
Figure 3.15: Tree Error Generalization.....	37
Figure 3.16: Model Deployment Process Using Docker.....	39
Figure 4.1: Linear Regression R-Squared On $2^k - 1$ Feature Combination.....	42
Figure 4.2: Linear Regression RMSE on $2^k - 1$ Feature Combination.....	42
Figure 4.3: Recursive Elimination Impact of Features In Accuracy.....	45
Figure 4.4: Decision Tree R-Squared for UFS Features.....	47
Figure 4.5: Root Mean Squared Error In Decision Tree Regression.....	48

Figure 4.6: Best K-Fold Value in Random Forest Regression.....	49
Figure 4.7: Estimator Effect on R-Squared with Random Forest Regression.....	50
Figure 4.8: Estimator Effect on RMSE with Random Forest Regression.....	50
Figure 4.9: Best Estimator After Filtering in 1.5 times Interquartile Range.....	51
Figure 4.10: Feature Importance.....	53
Figure 4.11: Average User Conversion Per Business Service Category.....	54
Figure 4.12: GitHub Actions Preview.....	55

ABSTRACT

Performance issues in loading web page contents and developing them such way that these becomes start interacting with users in fastest possible time has been a massive challenge for businesses over the years. Any medium to large sized web application may have several web pages and each of them may have several performance issues. One can fix them all but in perspective of meeting deadlines and allocating budget it can be cumbersome. The main objective of this research is to measure the impact of those performance issues that are creating massive impact in terms of other websites based on visual progression and developing a Continuous Integration, Continuous Development tool as a part of GitHub Actions marketplace to recommend developers and companies on prioritizing issues across various pages within their public and private repositories. This project based research follows quantitative research methodology including data collection, exploratory analysis of data, selection of features in uni-variate and wrapper techniques as well as re-sampling through train-test-split and K-fold cross validation techniques in several regression algorithms. We have developed a REST API by using dotnet framework to serve it as a service. Finally, we developed a GitHub Actions by using JavaScript on Node.js runtime to preview predictions on prioritization in several pages like a recommendation tool. This research shows time to interactive, boot-up time and largest contentful paint are the most important metrics in terms of boosting a site's visual progression based experience. This research is expandable in terms of other user experiences as well as the project in terms of authorization mechanism.

CHAPTER 1

INTRODUCTION

1.1 Background

Over the last decade till 2020 we've seen so many emerging technologies becoming the point of trust for various businesses worldwide. Among these trends the web still upholds it's necessity up to the mark. What we mean with the word necessity is that most of the emerging and currently trending new technologies are still highly dependent over the web. For example, the native mobile applications built with Kotlin or Swift or even the hybrid apps built with emerging technologies like React Native or Flutter are dependent on the web for communication. Over the time evolution of the web has been necessary for smooth user experience. Enterprise applications which are reaching it's consumers through the web are highly concerned about performance.

1.2 Motivation of the Research

Only responsive web page in various scaled devices isn't enough nowadays. I have been working with the web technologies and always been concerned about interactive designs as well as performance issues to enhance the web user experience. To do so I have used Lighthouse, an open source developer tool by Google comes with Chromium browsers. But the problem I faced is that whenever there are so many issues across different pages I often used to get confused which should I solve first that will enhance user's experience. After some research and discussion with other developers in the community I got to know that they are facing this problems too. So,

I thought of doing this project based research to find out the performance issues that impacts most in the real world user experience, specially in visual progression.

1.3 Problem Statement

In a web application there might be hundreds of performance issues in each page. Fixing them all can be time consuming. It's often confusing for project managers and web developers in making decisions about prioritizing performance issues across different pages that are creating impact on user's visual progression based experience. A recommendation tool is required that can suggest in each web page which performance issues needed to be prioritized to potentially save time and budget.

1.4 Research Question

- **Question 01:** What are the impact of web performance issues in visual progression based user experience?
- **Question 02:** What are the business benefits of conversion of a website to a progressive web apps?

1.5 Research Objectives

- To assist Web developers, Software Quality Assurance Engineers and Project Managers in making decisions on prioritizing performance issues based on visual progression across various pages.
- To assist Front-End Engineers setting up web best practices road-map while learning considering performance issues with prioritization.
- To help people from business background understanding the business benefits of web performance improvement based on case studies.

1.6 Research Scope

The research is focused on visual progression based user experience. Obviously, this is not the entire web user experience but this is a significant part of user's journey on a web page. The research is extensible in terms of predicting other user experiences like which are categorical in terms of their type. Currently, the project works on publicly available web pages. The project built based on this research can be improved with an authorization mechanism to help measuring protected pages too.

1.7 Thesis Organization

This thesis consists of 5 sections. Next sections include Literature Review in Chapter 02 based on different case studies, Research Methodology including feature selection, exploratory data analysis, model development and deployment using docker in Chapter 03. After that we discuss Result in Chapter 04. And we finally conclude with future works can be done from here in Chapter 05.

CHAPTER 2

LITERATURE REVIEW

Throughout this research we have performed some of the critical, constructive and comparative reviews on some of the relevant studies. In the next few sections we will cluster some our reviews over studies being performed on web performance and user experience by addressing their case studies.

2.1 Case Study on Web Performance Optimization

Throughout this study we are trying to demonstrate some of the recap in the domain of performance issues of the web. A novel approach of assessing the impression created on user experience by optimizing a website performance (Marang, 2018) proposed with few optimization techniques. They performed survey on some of the volunteers before optimizing their website and then they considered some of the optimization techniques including caching the static contents in Indexed DB associated with modern browsers nowadays, decreasing number of HTTP requests towards the application programming interface, reducing complexity of web workers, currently known as service workers and prioritizing the web contents preferred to demonstrate to the end users. They have seen an increment of 45% in their application introductory loading time and 18% improvement after caching in consecutively browsing the same web page. Their study shows even after optimization it didn't create significant impression on most of the users experience. Trevisan et al. (2019) developed a unsupervised from the traffic logs created by the users under internet service providers and they have made their tool open source. Their experiments result

shows web performance measurement metrics even in flow level are highly relevant with the site Speed Index. Ramakrishnan et al. (2020) proposed Page Loading Time (PLT) as the meaningful magnitude of performance optimization modeling. They have featured number of Document Object Model (DOM) elements along with the mean depth of the DOM including the JavaScript files or subsequent consecutive scripts in header or body of a web page which can either be synchronous or asynchronous in the preceding manner. Moreover, they have categorized the websites URL based on what content they produced to predict the PLT. Shivakumar (2020) has proposed a novel optimization framework as the modeled the preceding as maturity model. In the proposed framework they have prioritized mobile first design as per targeting devices of most of the end users. After that they have proposed a novel optimization life cycle like a software developed life cycle which is where they have prioritized architecture and mobile first design, coding, testing performance improvements and maintenance after A/B testing. Lastly, they've proposed some the optimization rules.

2.2 Case Study on Quality of User Experience

Assessing the web user's experience has been an emerging field for novel scientific research. From our previous case study we've got to know about web performance issues and new research trends for it's measurement. However, the goals and objectives of these measurements are due to improving user's experience to boost the businesses. Bocchi et al. (2016) have demonstrated a novel comparison of the time between existing performance measurement, Page Loading Time (PLT) for computing user's experience and the novel metrics proposed by Google in 2012 known as Speed Index (SI). They stated SI however wasn't well enough accepted at that time by

industries. They've figured out some of the possible reasons behind it like SI was computationally too much complex and it wasn't easy to understand at that time to perfectly measure user's experience. Initially, they've claimed PLT as not well acceptable due to Time to First Byte (TTFB) and Time to Last Byte (TTLB) is not good enough reflex for all possible servitude between rendering content and actual user experience. By using cumulative distribution over data collected from some top ranked websites, they've shown, SI lies in between TTFB and TTLB where the entire rendering and painting process is composed of TTFB and Time to Last Paint. In another study, Hoßfeld et al. (2018) have shown PLT was still widely used in that time to measure the experience quality from user's perspective and they tried to relate the SI with the Quality of User Experience (QoUE) in their study. Their point of discussion is based on Viewport, which is the visible part in the perspective of a user browsing a website based on the screen height and width. They however have proved that the PLT still computes even if the content that is not fitting inside a Viewport which is less relevant to user's experience whenever we talk about modern devices including android, iPhone or tabs. They however didn't exclude the PLT from measuring the user's experience but they proved that the measurement of visual Viewport interaction matters most upon measuring the quality of experience. They surveyed on 241 volunteers browsing experience and proved their claim of SI being significant by plotting SI with respect to Mean Opinion Score (MOS). Jahromi et al. (2018) have discussed a novel issue on measuring user's experience. They claimed issues in existing in PLT and SI is better with respect to solving problems due to PLT only. However, they proved that Viewport interaction based measurements are well accurate but it doesn't represent the entire user's experience.

2.3 Trends and Patterns Among Case Studies

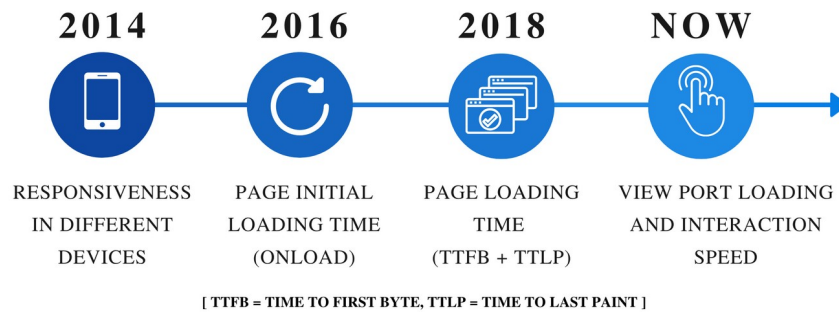


Figure 2.1: Measurement of Web User Experience Trends Over Time

In the figure 2.1, we can observe that measurement of quality of user's experience are changing over time like:

- During 2014, responsive sites were considered to have high user experience.
- A few years later, document and full page loading time including Time to First Paint and Time to Last Paint became popular to measure quality of experience.
- Nowadays, instead of full page loading, visual progression and view port's interactivity speed seems to be considered most important to measure quality.

2.4 Addressable Improvements

Based on all the studies and review above now we will address some of the areas which shapes the direction toward novel research scopes. So far we have got to know about various studies which is where most of the works has been conducted practically in terms of responsiveness, best practices and document initial loading speed. The problem with this is that we still cannot decide what the performance issues actually important while it comes saving project budget and time. So to solve this we need a novel approach of measuring the variability of performance issues as well as the most important issues responsible for degrading user's experience on a

web page. To be more precise, we have noted some improvements required based on current best metrics of user experience measurement. The table below demonstrates improvements required based on modern web architecture considering single page application built with Angular, React.js or Vue.js as well.

Table 2.1: Addressable Improvements Figured from Case Studies

Improvements	Justification
Page or content loading time cannot be the entire representative of the quality of experience of a site.	Page loading time includes loading time of those contents too which are still not visible and out of user’s interaction area.
Measurement of quality of web user experience needs to be independent of Document Object Model depth or size.	Modern websites built nowadays using Webpack can have large DOM depth but high in performance.
There are hundreds of performance issues but these requires prioritization for saving time and budget for project.	Improvement requires cost and time to achieve and for enterprise or startups cost optimization is so important.

CHAPTER 3

RESEARCH METHODOLOGY

Our proposed research and development methodology starts with understanding the business augmentation by analyzing the business success trends of companies that focused and spent their budget on improving performance. Consecutively following the analysis we've decided the required data we need for our research work to conduct. Afterwards, we have collected our required data by using Page Speed REST

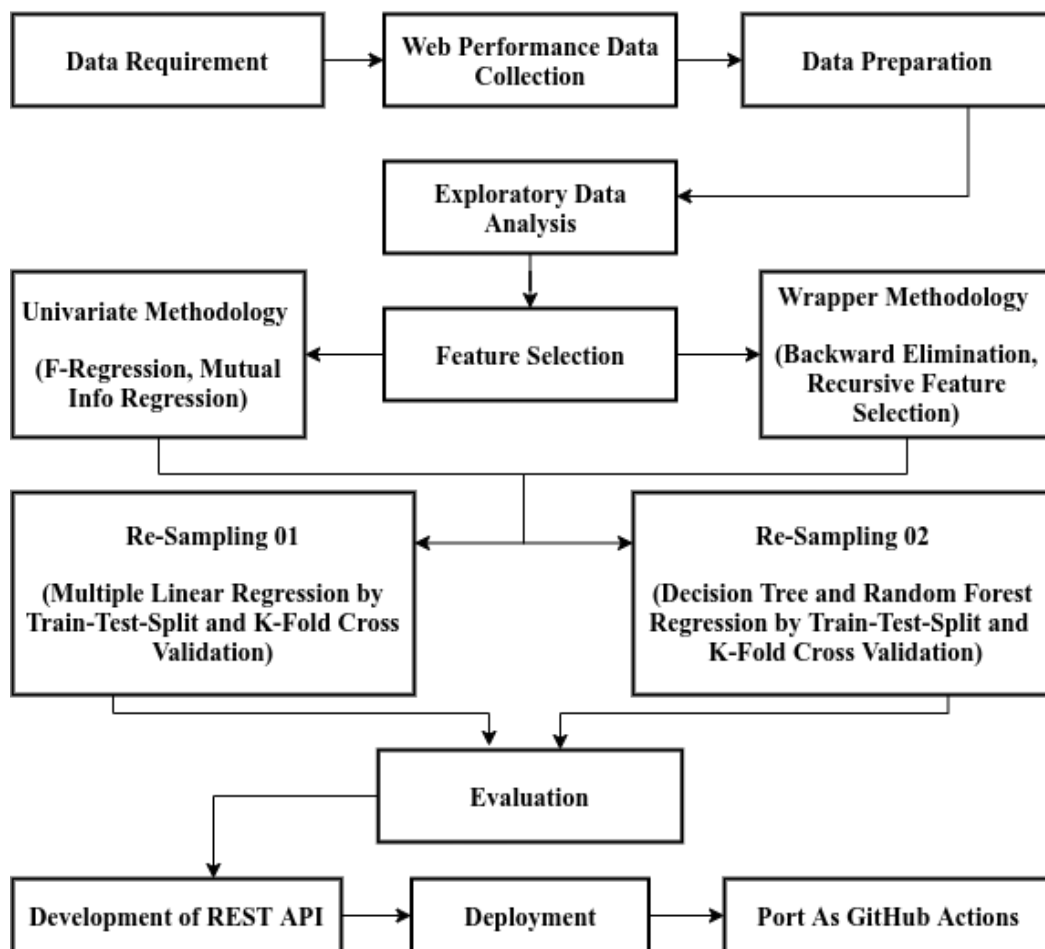


Figure 3.1: Proposed Research and Development Methodology

Application Programming Interface (API). Then we have conducted some Exploratory Data Analysis (EDA) on our collected web performance data to understand what we can do further with our experiments. To avoid redundancy we have conducted our test driven feature selection. This step has been the most critical and lengthy process throughout our research work. Afterwards, we have evaluated the training results from our feature combinations and elected the best model for our further works. In this phase, our research work is ready to be started for implementation. So, we have developed a Representation State Transfer (REST) API. After implementation of our API, we have deployed it to cloud as a service.

3.1 Data Requirement

Our research work starts with making decision on what data we need to conduct further research tasks. In this step, we have to use the insights we have found from our case study reviews. In our case study on web performance, we have got to acknowledge about sever performance issues can happen on an end users device. In our another case study on User Experience (UX) we have got to know about different measurements of UX getting trending over time. So based on our previous studies we are deciding to collect these information:

- For proper UX measurement, firstly, we need make a list of websites that people visit most based on trending topics.
- Afterwards, for our listed websites, we have to collect their websites performance issues existing and these are going to be our features.
- Based on our UX study in Section 2.2, we will collect the Speed Index (SI) of our listed sites and this is going to be our target variable. Because, in most of the studies SI is the most acceptable metrics for addressing visual progression

based user's experience. So, we will try to find the best suited model that fits best for predicting user's experience.

- However, our primary goal is to find the feature importance of our best model. Because, our primary objective is to measure the impression of performance issues on visual progression based user's experience.

3.2 Data Collection

From our data requirement understanding step we got to know about the data we require to collect. So, first we used Google's Programmable Search Engine to find the sites with high traffic based on several trending search categories.

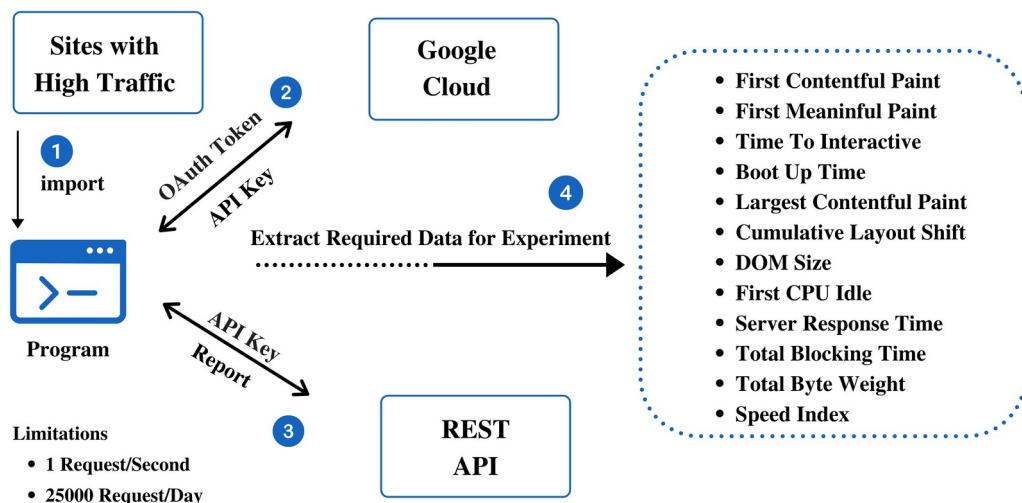


Figure 3.2: Data Collection Process from Page Speed Insights REST API

Firstly, We've used Google Search Trends to find out most important search categories worldwide. Then we have listed what type of keywords we are going to use to develop our URL list. In the following table, we listed the popular categories our URL collection sites serve their consumers.

Table 3.1: Categories of Sites for Which Performance Reports Are Collected

URL List Categories	Example
Business	Import, Export, Stock Exchange, E-Commerce etc.
Entertainment	Movies, Television Drama, Web Series, Cartoon etc.
Healthcare	Diet, Exercise, Pandemic etc.
Science and Technology	Space, Machine Learning, Artificial Intelligence etc.
Sports	Football, Cricket etc.

The most critical part of our data collection process was to keep API rate limitation in consideration. To collection performance issue reports we have used the Page Speed Insights (PSI) REST API by Google which is where:

- We had to first authorize our Google Account first by using the token provided by Google. We used our token to authorize for using PSI API.
- Afterwards, for each 25,000 URLs in our collection we've tried to run our program in an asynchronous manner each day. Because, According to the PSI documentation we can only hit the API 25,000 time in a day.
- For our 100,000 URLs, it took almost 4 days to complete the Data Collection process. We have gathered each day collection in separate Comma Separated Value (CSV) files as it is well accepted format for Data Science experiments.
- Because, we conducted our experiments in Google's python environment also known as Google Co-Laboratory, we kept our files in Google Drive.

Table 3.2: Data Attributes and Details

Attribute	Description	Type
FCP	First Contentful Paint	Numerical
INTERACTIVE	Element Interaction Time	Numerical
SRT	Server Response Time	Numerical
DOM_SIZE	Document Object Model Size	Numerical
BUT	Boot Up Time	Numerical
FMP	First Meaningful Paint Time	Numerical
TBT	Total Blocking Time	Numerical
TBW	Total Byte Weight	Numerical
FCI	First CPU Idle Time	Numerical
CLS	Cumulative Layout Shift	Numerical
LCP	Largest Contentful Paint	Numerical
FID_CATEGORY	Category of FID	Categorical
LCP_CATEGORY	Category of LCP	Categorical
CLS_CATEGORY	Category of CLS	Categorical
FCP_CATEGORY	Category of FCP	Categorical
UX_INDEX	Speed Index of a site. We used it as our visual progression base UX score.	Numerical

3.3 Data Preparation

Before starting our Data Analysis, we prepared our data to avoid noise while exploring web performance issues and experience data. It is not a novel but an important step before conducting EDA. Our data preparation tasks are as follows:

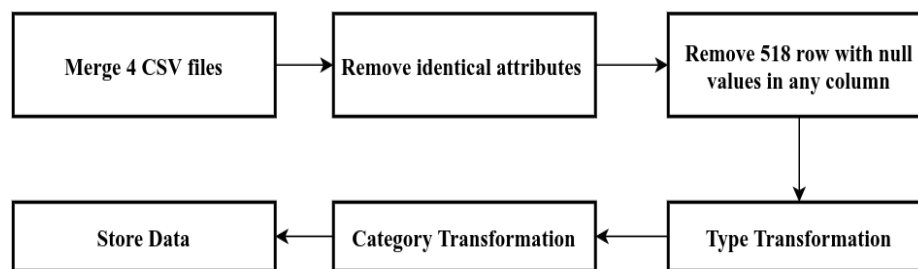


Figure 3.3: Data Preparation Steps

As discussed previously in the Data Collection section, it took four days to collect our entire data set as per consideration of API rate limitation. So, our initially collected data are in structured format stored in 4 separate Comma Separated Value (CSV) files. To conduct further experiment, we need to merge the entries like:

- At first, in our data preparation step we have merged those four CSV files.
- We found 2 of our attributes section having less variance in their values and most of them were null. Those two attributes are Unused JavaScript and Unused Styles. So, to avoid noise in analysis, we removed them.
- Afterwards, we explored for entries for which some columns carrying null values anyway. To avoid further redundancy we removed those **518 entries** which are almost **0.661%** of our initially collected dataset.
- As most of our performance metrics are numerical, so we transformed them into float types and categorical information to numbers as follows: FAST = 1, NEED IMPROVEMENT = 2, SLOW = 3 and stored finally.

3.4 Exploratory Data Analysis

Before our feature selection and modeling, we focused on learning from our collected web performance report. To achieve this goal, we first conducted Quantitative Exploratory Data Analysis (EDA) on our performance report. At first, we tried to explore our issue report data distribution by using Mean and Median formula. Mean of each web performance issue is calculated as follows where n is the total number of entries we have in our dataset and i is addressed for iteration:

$$MEAN(Performance\ Issue) = \frac{1}{n} \sum_{i=1}^n issue_i \quad (3.1)$$

By doing summation of each performance issue entries addressed as i -th iteration in the equation 3.1 and then dividing by total occurrence we got mean for each of our features as follows:

Table 3.3: Mean of features and target

Attribute	Mean	Attribute	Mean
FCP	949.896877	TBT	376.275662
INTERACTIVE	4166.749076	TBW	3.202074e+06
SRT	953.214688	FCI	3384.799288
DOM_SIZE	1322.833920	CLS	0.173483
BUT	1906.515876	LCP	2554.590280
FMP	1128.364308	UX_INDEX	3008.959377

As mean of performance issues can be influenced by outliers, so, to reduce the impact of outliers, we have also calculated median to see our data distribution which is also

sometimes referred as 50th percentile of data where n is the total entries and it is even.

So based on this the median calculation is being done with the approach below:

$$\text{Median}(\text{Performance Issue}) = \frac{\text{Issue} \left[\frac{n}{2} \right] + \text{Issue} \left[\frac{n+1}{2} \right]}{2} \quad (3.2)$$

By calculation of median, we can see extensive variance among few features. Our target variable seems to have huge difference in comparison with it's mean value which is demonstrated below:

Table 3.4: Median of features and target

Attribute	Median	Attribute	Median
FCP	829	TBT	150
INTERACTIVE	3278	TBW	2.168678e+06
SRT	761.57	FCI	2711.5
DOM_SIZE	917	CLS	0.06
BUT	1049.827	LCP	2055
FMP	943.50	UX_INDEX	2657.77

Our observation from mean and median of performance issues demonstrates there are possibilities of existence of outliers. Before model training, most of the data scientists prefer removing outliers. But there are certain cases where we can remove them and where we cannot. Our point of justification for not removing outliers are as follows:

- We have collected our data entries by using PSI API which is where it's main origin of source is Google Cloud.
- Chrome User Experience Dashboard documentation, they collected user experience data with the consent of users through Chromium Based Browsers.

- So, our entire data set contains information which are collected computationally in real time.
- Outlier removal is suitable when data is aggregated by human interaction. For example: Collecting survey data from school going kids might include erroneous information.
- In another case, outlier removal is suitable when the data collected is some sort of impossible value. For example: A kid might enter his or her age as 400 while survey just for fun. But in our case, all the feature values are computationally calculated by actual performance issues and visual progression where they can be in a range of $0 \leq n \leq \text{Infinity}$ as per Page Speed Insights documentation.

Upon above observation and discussion of our initial numerical EDA, we have decided not to remove outliers because in our case for several performance issues and visual progression based on different network and device performance condition they are just extreme data point values not always error prone. Our next iterative process of numerical EDA is about finding the relative importance between features and response variable as well as relative importance between performance issues. Our goal of performing this EDA is to boost our next step which is feature selection. To do this numerical EDA, we have chosen Pearson Correlation Co-Efficient (PCC). To understand how this works in our research we have to understand three formula consecutively. The first things comes while calculation of PCC is variability of our data points. Let's say, we want to find the variability of our feature data INTERACTIVE. To be more precise, we want to know how much INTERACTIVITY

among sites are spreading in our collected data. To achieve this goal, we do the mean of squared distances of the feature's data point by using the formula below which is where n is the number of entries in our dataset, issue addresses to performance issue, currently INTERACTIVE in our case and V_p is variability of performance:

$$V_p = \frac{1}{n} \sum_{i=1}^n (\text{issue}_i - \overline{\text{issues}})^2 \quad (3.3)$$

The problem with V_p is the unit of main performance issue is not remaining the same. For INTERACTIVE, it get square of second. So, we need to find the root of V_p which is where it is known as Standard Deviation, S_p as follows:

$$S_p = \sqrt{V_p} \quad (3.4)$$

Now this is just, one feature variability or spread. But our goal is to find how much each feature performance issue is relevant in terms of user experience. To achieve this we conduct another operation which is known as co-variance, in our case, we are trying to find the feature and response variability together. For example:

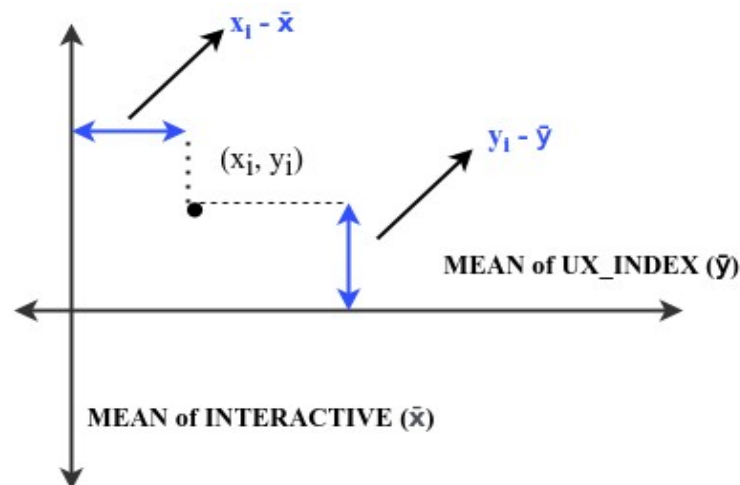


Figure 3.4: Variability of Interactivity and User Experience

In figure 3.4, we've tried to demonstrate how a feature and response variability occurs in terms of performance and experience. For any set features, in this case, INTERACTIVE is assumed to be addressed as x and for experience set, UX_INDEX is referred as in y-axis. In those axis, we've drawn the line of mean and for any point of x_i and y_i we have tried to figure out the distances together mentioned as a point in the graph. In this graph, we've tried to figure out the combined variance and this is some point which is where we have found our desired co-variance. This equation is mentioned as follows where $CoV_{(p, e)}$ is the co-variance of performance and experience, issue refers to performance issue and experience refers to user experience based on visual progression. The equation is as follows:

$$CoV_{(p, e)} = \frac{1}{n} \sum_{i=1}^n (issue_i - \overline{issues})(experience_i - \overline{experiences}) \quad (3.5)$$

Based on equation 3.3, 3.4 and 3.5, we finally calculated Pearson Correlation Coefficient (*PCC*) on our collected dataset for finding relevance between features and relevance between features and response variable. By applying Co-Variance, Standard Deviation and Variance our final equation based on *PCC* stands for:

$$PCC_{(p, e)} = \frac{CoV(p, e)}{(S_p)(S_e)} \quad (3.6)$$

$$PCC_{(p_1, p_2)} = \frac{CoV(p_1, p_2)}{(S_{p_1})(S_{p_2})} \quad (3.7)$$

Equation 3.6 states *PCC* between features and response where 3.7 states *PCC* between features, in our case, performance issues. In equation 3.6, S_p and S_e are the standard deviation for performance issues and experience consecutively. In equation 3.7, $CoV(p_1, p_2)$ is the co-variance between any two features where S_{p_1} and S_{p_2} are

addressed to notate the standard deviation for any two performance issues respectively. In any case of circumstances while measuring co-efficient we consider:

- For any *PCC* between performances or performance and experience values that are close to positive one (+1) are considered positively highly correlated and values close to negative one (-1) are considered as negatively correlated.
- For any *PCC* if value is zero or very close to zero then the relation is considered as having no correlation.

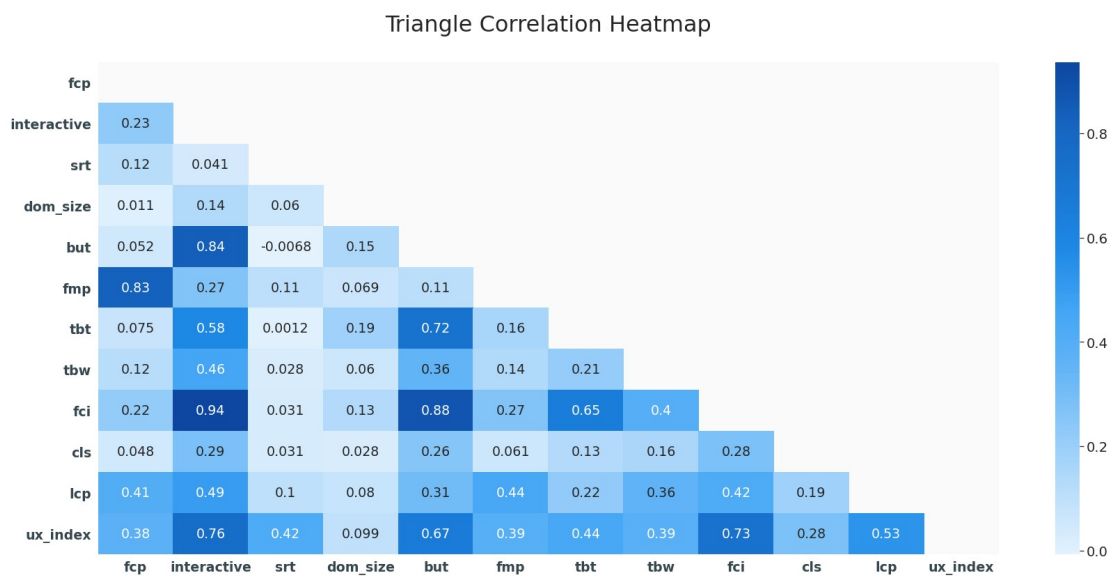


Figure 3.5: Pearson Correlation Co-efficient Among Attributes

From this numerical EDA, we have observed few variability of correlation among features and response variable, in our case, UX_INDEX. We have observed:

- INTERACTIVE, FCI and BUT having high correlation with our response variable, UX_INDEX. This means, these performance issues have high significance with respect to visual progression to end user's device.
- DOM_SIZE and CLS are less significant in terms of linear relation with UX_INDEX. But that doesn't mean they don't have any contributions towards

calculation of UX_INDEX. Because, PCC just shows relationships based on linearity between two variables.

- Among features, we can see that INTERACTIVE and FCI are highly correlated. Which means when a site elements are ready to interact with end users, CPU of the device starts reducing it's usage of resources.

To observe more details in between relations between features and target, we have prepared some statistical plots demonstrated below:

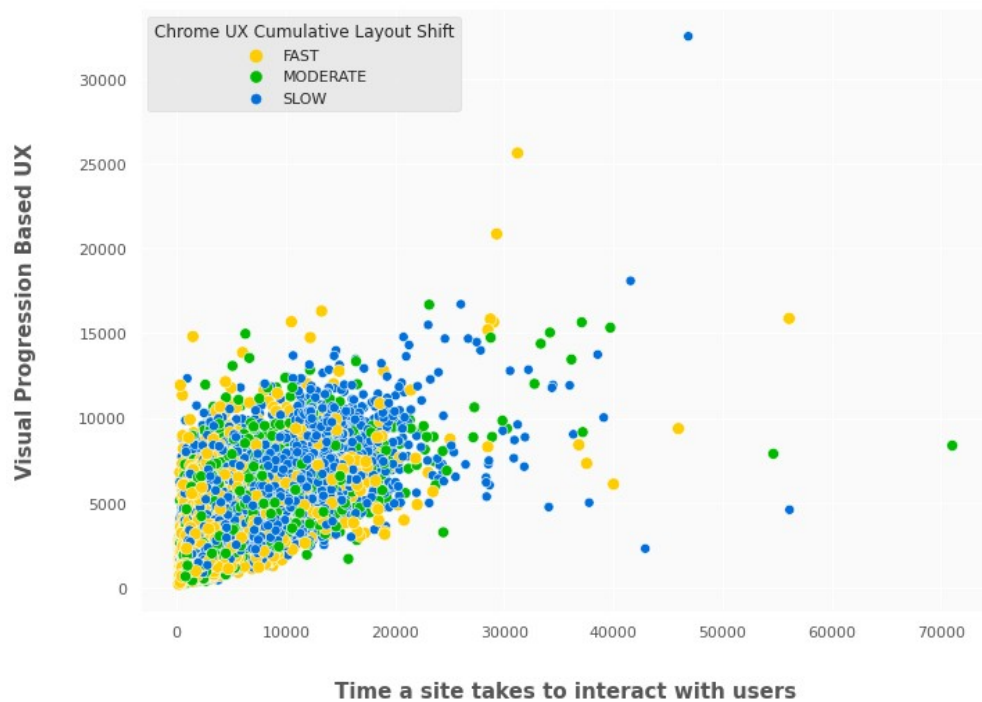


Figure 3.6: Layout Shifting based Correlation (Interactive & UX Index)

In the plot above, we observed the distribution of correlation between INTERACTIVE and UX_INDEX in terms of one of the Chrome UX Reports Cumulative Layout Shift. The plot demonstrate site elements that takes less time to interact with their end users are expected to have better cumulative layout shift on end

users devices, more specific, in mobile devices. When INTERACTIVE and UX_INDEX both increases in linearity, layout shift gets worst on users end which can be frustrating for a user. We also observe that our user experience data collection has less skewness in comparison with layout shifting.

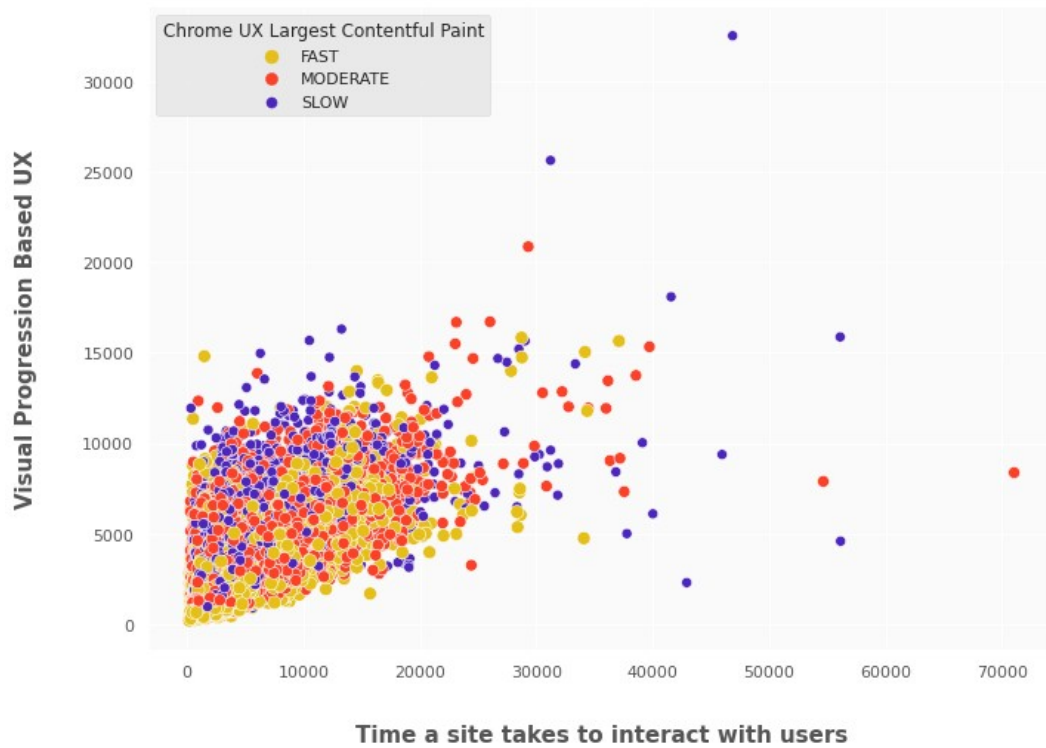


Figure 3.7: Largest Content based Correlation (Interactive & UX Index)

Another of our numerical EDA on the relation between INTERACTIVE and UX_INDEX based on their distribution of time the sites take to stream or visualize largest content is being demonstrated in Figure 3.7. This report clearly shows sites with reasonable UX_INDEX are most likely to make their largest dom content to be visualized or streamed faster. This is clearly shows that our case studies in section 2.2 on user experience is now not that critical as it was to be before 2018. In terms of

largest dom elements taking time to be paint is leading toward visual progression to be affected as long as takes time to initiate on end user.

In another of our EDA, we have tried to observe the distribution of our response variable, in this case, UX_INDEX. The distribution is demonstrated below:

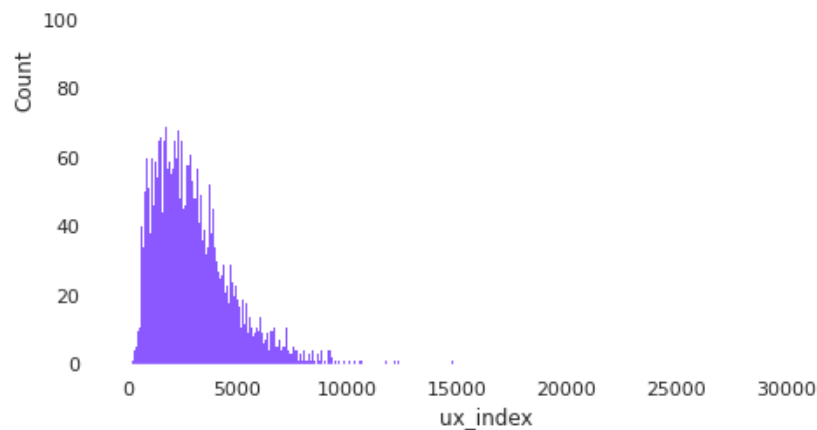


Figure 3.8: Distribution of Visual Progression

According to Google this measurement is scaled as itemized below as per PSI documentation:

- **Good Visual Progression:** $0 \leq \text{UX_INDEX} \leq 4300$
- **Moderate Visual Progression:** $4300 < \text{UX_INDEX} \leq 5800$
- **Bad Visual Progression:** $5800 < \text{UX_INDEX}$

In our dataset, most of the sites are having Good to Moderate visual progression. Also, there is good amount of entries with bad visual progression. So, from here we can conclude that our data distribution for UX_INDEX is having good variability. To understand the distribution of UX_INDEX more precisely we want to see how much Layout shifting is happening in real world end user's devices. The major difference

between CLS and CLS_CATEGORY is that the first one is lab data calculated by PSI API. And second one is the actual CLS happening in end user's device but addressed in a categorical manner. In the figure below, we have found more insights out of our UX_INDEX data points distribution in terms of real user aggregated CLS_CATEGORY collected throughout chromium based browsers, for example: Chrome, New Microsoft Edge, Brave etc.

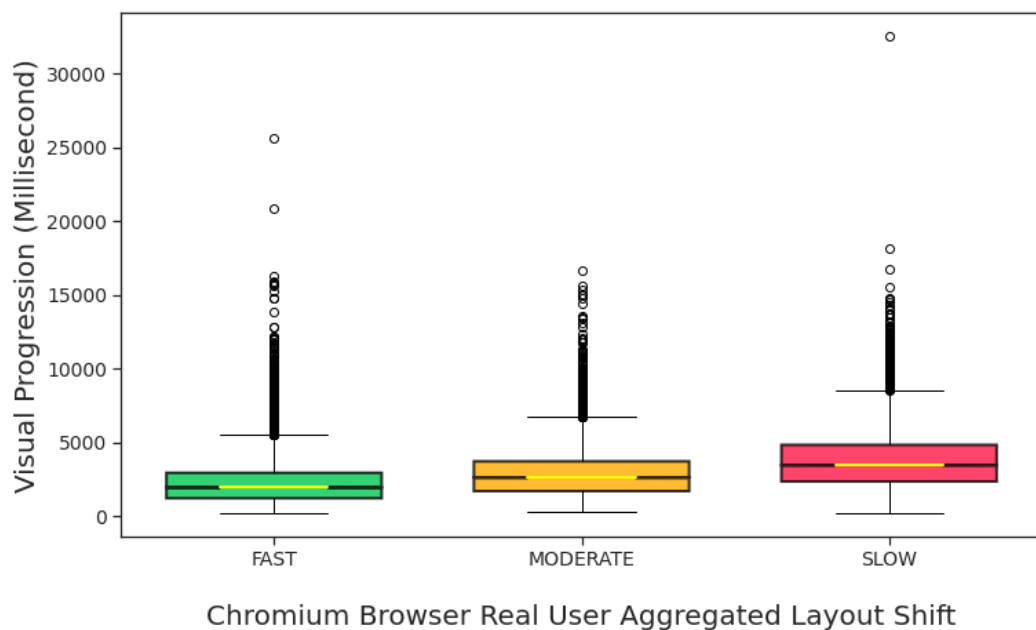


Figure 3.9: Distribution of Visual Progression Based on Layout Shift

From the above EDA, we can see that sites that take less time to visually progress are most likely to have faster layout shifting. As per Google documentation, visual progression under 4300 milliseconds are likely to provide best experience to their consumers. This is also clearly states that Slow Layout Shifting has large variance in our dataset. One thing we must conclude that the values out of Interquartile Range (IQR) and Upper whisker of the plots are not outliers. Because, according to PSI documentation these value can be infinite in upper limit. So, we can explain them just

as extreme data points small in number. It can be balanced by increasing the amount of extreme values or decreasing the amount of value that is inside IQR.

In our final EDA we want to see the distribution of UX_INDEX in terms of Largest Contentful Paint aggregated from chromium based browsers with user consent:

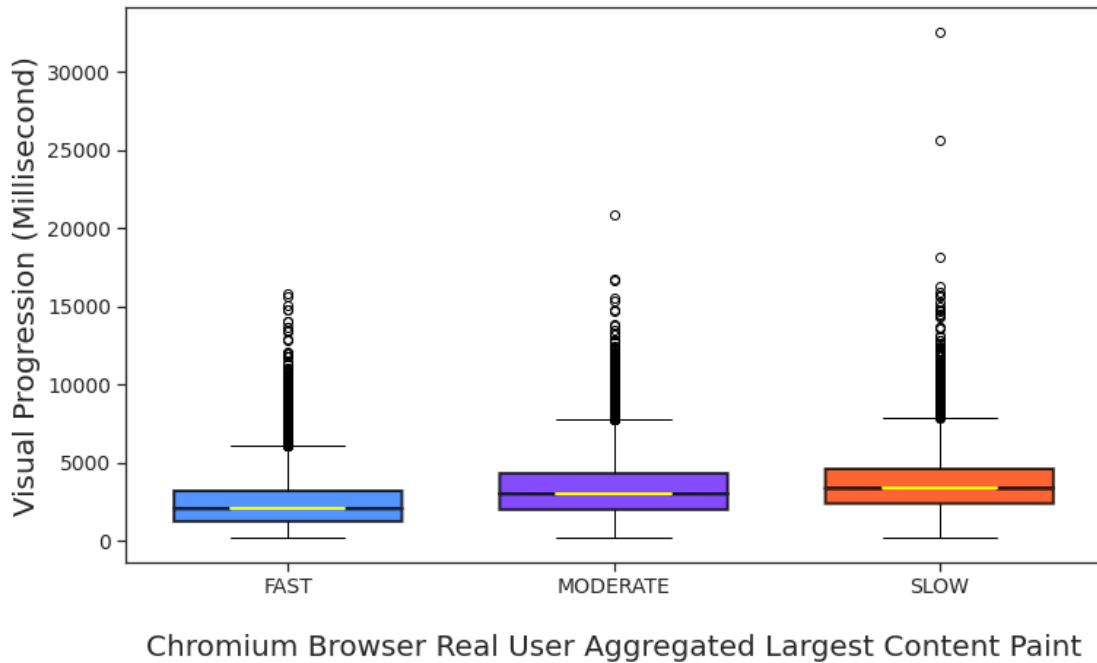


Figure 3.10: Distribution of Visual Progression Based on Largest Content Paint

In this plot, we see larger distribution variability of LCP in moderate speed. It also states that sites with sites that have fasted visual progression are most likely to serve their largest contents faster. There can be two possible reasons:

- The largest content they load on their end user’s device are extremely small in size. For example: Compression of images or video quality based on network condition etc. which is a best practice too according to our case study 2.2.
- In these sites, largest contents are being served as per modern browsers best practice, more precisely, images in WebP format or, videos in WebM format.

We've got to know about the relation in between performance issues and experience as well as distribution of progression in terms of aggregated UX. Finally, we conclude that our target UX_INDEX is a pretty good measurement of user experience.

3.5 Feature Selection

Before applying machine learning techniques, we need to find for which set of performance, UX_INDEX can be best predicted. Because, the robust and sustainable model we can develop to predict UX_INDEX, the more we can find the impact outline of performance issues on visual progression. Our approach for selecting features has been achieved in two popular techniques:

- Univariate Feature Selection
- Wrapper Methodology

Our features are numerical as well as our response variable UX_INDEX. So, the problem is a **Regression Problem**. One thing we want to stand out from here that our main objective of doing feature selection is to find the strongest set of performance issues relationship with visual progression respectively. First we have done Univariate Feature Selection which is where we do not collectively look over performance issues. Instead, we isolate each of the performance issues to observe their relation significance with the visual progression based user experience. This process is repeated for all the performance issues by sending each of them to isolation and test their significance. The advantage of this process is that we can particularly look at each of the performance issues which is where we can find out their linear significance. Our first Univariate Feature Selection (UFS) standard with F-Score based test driven feature selection. Our main goal of doing F-Score based UFS is to find the k-best significant features by ranking the significance of features with respect

to user experience. We will further discuss about the workflow in the algorithm section. To capture the non-linearity we have tested with another UFS, named as mutual info regression comparing with user experience. Our feature selection process is as follows demonstrated below:

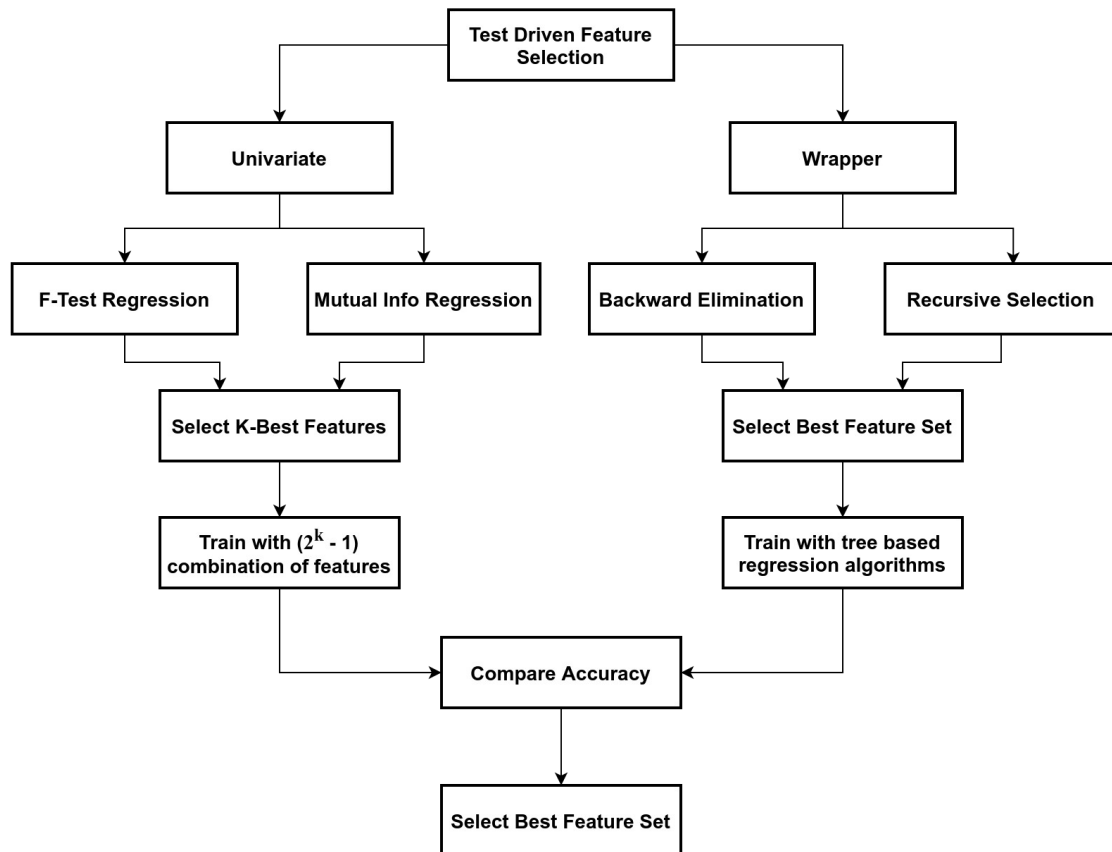


Figure 3.11: Test Driven Feature Selection

To find more accurate results, we have performed another Feature Selection, which we call Wrapper Feature Selection (WFS). Then by comparing both ranking we have extracted out most significant performance feature set to predict experience. Then we have applied Multiple Linear Regression with two re-sampling techniques, Train-Test-Split (TTS) and K-Fold Cross Validation (KCV). In another feature selection we have used Backward Elimination Techniques and Recursive Feature Elimination

Techniques to find the best feature set based on tree algorithms. Then after testing with TTS and KCV in tree based algorithms we have compared both results of WFS and UFS. The algorithms we have used in this research will be discussed in next section respectively covering both UFS and WFS techniques.

3.6 Algorithms Used

In this section we will discuss each of the algorithms we have used in Univariate Feature Selection and Wrapper Feature Selection throughout model training. We will have some short overview on how performance issues and experience fits in these algorithms to solve regression problem.

3.6.1 F-Test Regression

Our main objective of doing this test driven regression is to have brief comparison of our regression analysis to demonstrate the variability with respect to user's visual progression based experience. We start doing F-Test by considering a restricted model in predicting the accuracy of experience. Let's say, from PCC, we found *INTERACTIVE* to be the most significant predictor of *UX_INDEX*. So, our restricted model starts with the simple model demonstrated through equation below where *UX* refers to *UX_INDEX*, *m* refers to co-efficient and *c* is any constant:

$$UX = m * INTERACTIVE + c \quad (3.8)$$

Our initial goal of F-Test is that for each feature in our dataset except *INTERACTIVE* we will test their significance and rank them as well. For each feature to be ranked we start with two hypothesis:

- **Null Hypothesis, H_0 :** Newly constructed model with the performance issue cannot properly explain the variability of user experience.

- **Alternate Hypothesis, H_1 :** The model constructed with current performance issue is a significant dependent variable in terms of predicting user experience and it is highly acceptable for further experiment. This process is repeated as a part of iteration for all the f-test in terms of linearity.

Our primary objective of doing the above two hypothesis is that from valuable significance of performance we want to compute that the H_0 can be accepted or rejected in manner. Instead of considering whole as a process we will look what happens in a single F-test in terms of find significance of the performance issue. The steps are as follows:

- **Intuition Construction:** Let's say, we want to add FCI in our model as another predictor of experience in the equation 3.8. So, we address our initial model as M_i and our currently model with FCI is addressed as M_c . Let's say, we have k number of entries in our dataset. Let's say, model M_i has p number of features and model M_c has q number of features. Let's say, the residual of sum squares for model M_i is R_i and residual of sum squares for model M_c is R_c .

Based on this the F-Statistics is calculated as follows:

$$F \text{ statistics} = \frac{\left(\frac{R_i - R_c}{q - p} \right)}{\left(\frac{R_c}{k - q} \right)} \quad (3.9)$$

- **Density Function:** In the next step, we try to find probability distribution function of the F Statistics in terms of performance and experience. After all those computations, we try to decide, if we can reject the null hypothesis in terms of predicting user experience. This test driven approach finally returns a ranking of features from where we decided keeping K-Best features.

3.6.2 Mutual Information Regression

The problem with F-test is that it ranks features with only linearity while in our case it tries to find K-Best performance issues that are having significance in terms of predicting visual progression. How it works is that:

- This algorithm also tries to find K-Best features like F-Test but the advantage of this algorithm over F-Test is that it considers non-linearity while calculation of UFS. So, in our case, it tries to find non-linearity relevance for performance issues with respect to experience on end users device.
- It calculates how a performance issue is mutually dependent with users visualization based experience. Finally, after all calculation it returns a ranking of performance issues from which we decided to select feature set for further experiment in comparison with F-test and reducing feature standpoint where in both cases less significance are not considered.

3.6.3 Multiple Linear Regression

To understand how Multiple Linear Regression (MLR) works in our case, we first have to understand how Simple Linear Regression (SLR) works.

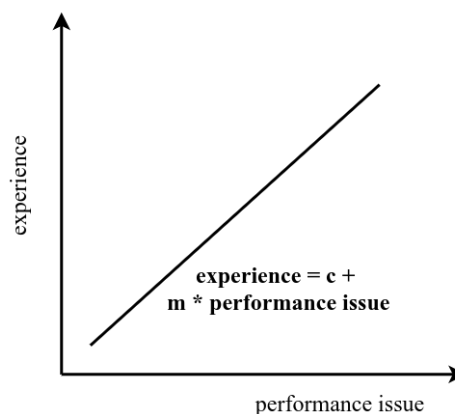


Figure 3.12: Linear Regression

What happens here is we try to find the slope (m) which is the tangent of the regression line with horizontal axis and intercept (c) to measure the impact of a single performance issue with our response variable UX_INDEX. First we assume that we have found the best fitted line to predict UX_INDEX in terms of a single performance issue. Then we try to find how well we can predict UX_INDEX with that performance issue. It is the distance from the performance issue's data point towards the fitted regression line assumed earlier. This error is the residual error of our performance line. For all performance data point we tried to find the mean of all those residual errors which shows how poorly the performance issue line fits with whole UX_INDEX. This error is called Mean Squared Error (MSE). Our goal is to find the best line of performance issue for which MSE is optimized and UX_INDEX can be best predicted. Let's say, for any performance issue P our collection for independent variable is P_i . In case of our response variable which is UX_INDEX, U , we call it U_i . To calculate the best and optimized co-efficient, we used the following equation:

$$m = \frac{\sum_{i=1}^n (P_i - \bar{P})(U_i - \bar{U})}{\sum_{i=1}^n (P_i - \bar{P})^2} \quad (3.10)$$

From this we can calculate the intercept too. This is just a SLR. In multiple linear regression we consider all our performance issue which are being selected best feature set from Univariate Feature Selection to fit enough to predict UX_INDEX. Our goal of doing Multiple Linear Regression here is to identify the strength of the impact of performance issue on user's visual progression. For UX_INDEX, U , we can summarize the whole MLR equation as follows where m is the co-efficient, P is the performance issue and c is the intercept:

$$U = c + m_1 P_1 + m_2 P_2 + m_3 P_3 + \dots + m_n P_n \quad (3.11)$$

To simply the MLR equation in a Vector form we can address it as follows:

$$\hat{U} = m^T P \quad (3.12)$$

Here we have expressed the visual progression based user experience as the dot product of two vectors m^T and P , respectively representing co-efficient and performance issues.

3.6.4 Tree Based Regression

To infer the users experience in terms of performance issues, we have explored two other algorithms known as Decision Tree Regression (DTR) and Random Forest Regression (RFR). Our main goal of trying tree based regression is to adoption of non-linearity between performance issues and experience. The advantage of tree based regression algorithms over our previously tried algorithms is that we don't need to scale the performance issue data points. While experiment with these algorithms our main target was to ensure the reduction of Generalization Error (GE) as much as possible. Let's say, there is a respective mapping of performance issues P towards user experience U mentioned where f is the mapper as follows:

$$U = f(P) \quad (3.13)$$

Here our main goal is to find a mapper f' which is where we try to find the best prediction of the initial mapper f where we want to eliminate noise at best. While doing experiment with tree based regression we tried to overcome two problems.

Those problems are as follows:

- **Over fitting:** Novel mapper f' unnecessarily fits noise of performance data points during training. The problem with this issue is that it extremely vanquishes it's power of predicting experience data points out of sample.

- **Under fitting:** Another issue while reducing GE is to cop up with under fitting issue. Here trained model isn't enough flexible to capture how performance issues relevant enough to predict experience.

While reducing GE, we also have to find out the point of complexity of our model (MC) where it best fits as well as the GE is less enough. In tree best regression, MC can be constructed well enough by reducing GE. To do this, we try different set of tree depth and minimum samples to be kept in each leaf. What we mean is:

- **Minimum Tree Depth (MTD):** Setting up flexibility level upto where we want our fitted regression line to capture non-linearity during prediction of user experience in terms of performance issues. We address it as the MC in our case. The greater the MTD is, the greater the MC will be fitted.
- **Minimum Samples Per Leaf Node (MSL):** Number of performance issue set of data points in training set we want to keep in each leaf node.
 - While this value is integer, it keeps that number of samples of performance issues while training.
 - In case of the value being fraction, it calculates the percentage of total performance issue training set and keeps while training. For example: When we define this value as 0.1, it keeps 10% performance issue sample in each leaf node in terms of predicting experience.

After considering all those criteria, we started finding actual GE. To measure this we do this as follows:

$$G_e = b^2 + v_a + i_e \quad (3.14)$$

Where in 3.14, G_e is the Generalization Error, b is the bias (On a mean how different f and f' are), v_a refers to variance and i_e refers to irreducible error as well. We will further discuss about this in evaluation section.

3.7 Re-Sampling

We have discussed earlier that our approach of experiment was test driven model development. Generally, computation of accuracy of the model and reducing error are the main concern while enhancement of training in data science experiments. For this purpose, we have applied two specific Re-Sampling techniques in both of our Linear Regression algorithm and Tree based regression algorithms. These two techniques are discussed below in terms of enhancement of model.

3.7.1 Re-Sampling by Train-Test-Split

In Train-Test-Split (TTS) re-sampling technique, we divided the dataset into two parts. The larger part (70-80%) is called training dataset and the smallest part (20-30%) is test dataset. This is demonstrated in the figure below:



Figure 3.13: Train Test Split Re-sampling

Our goal of doing this re-sampling is to enhance user experience prediction and evaluation on out of sample performance-experience dataset. As we decided to develop a REST API with our final model, so we need to make sure that our model

returns as much as possible accurate results in a real world environment as a plugin or Continuous Integration and Continuous Development tool.

3.7.2 K-Fold Cross Validation

The problem with TTS is that in terms of predicting user experience from performance data points it will suffer from high variability and it is highly dependent on it's dataset. This inconsistency leads us to a problem which may be a point of failure in real world impact measurement.

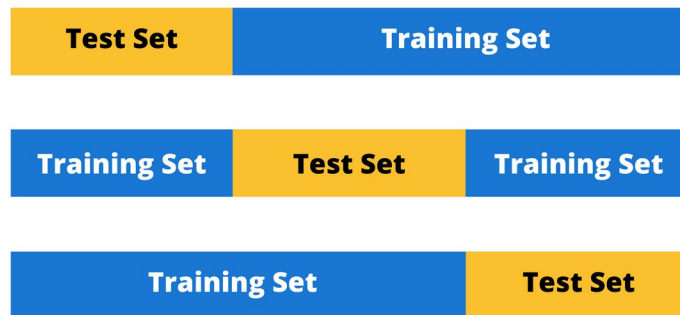


Figure 3.14: K-Fold Cross Validation

Above figure demonstrates how K-Fold cross validation works. Here we use our entire performance-experience dataset to perform multiple TTS. The advantage of doing this over TTS is that:

- The variability of performance issues in predicting experience while TTS is significantly normalized now.
- Out of sample accuracy in prediction of users experience is highly consistent in comparison with the consistency of TTS.

Our primary goal of working with re-sampling techniques is to enhance the model performance and consistency in prediction of user experience. Though our main goal

of doing this research is not measuring the UX, still we want the model to be consistent enough to get the best feature importance.

3.8 Evaluation and Generalization

Our final step before development of REST API is to evaluate and compare the models generated in the steps by UFS and WFS by using the re-sampling techniques used mentioned above. To do this, we have worked with several evaluation techniques that measures error raised by prediction result.

3.8.1 Mean Absolute Error

In statistical and our research relative terms, Mean Absolute Error (MAE) is the error between coupled entries addressing the same occurrence. It is measured by using predicted UX (UX_p) and actual UX (UX_a). For k samples we denote the calculation formula as follows:

$$MAE = \frac{1}{k} \sum_{actual=1}^k |UX_a - UX_p| \quad (3.15)$$

3.8.2 Root Mean Squared Error

Earlier when we discussed about Residual Errors in MLR section, we measured distance of performance issue from the fitted regression line. When we calculate mean of all errors being caused residual for actual UX (UX_a) and fitted line of UX (UX_f), we denote Mean Squared Error (MSE) as follows for k samples:

$$MSE = \frac{1}{k} \sum_{i=1}^k (UX_a - UX_f)^2 \quad (3.16)$$

To calculate Root Mean Squared Error (RMSE) we calculate as follows:

$$RMSE = \sqrt{MSE} \quad (3.17)$$

Goal of doing RMSE is that we want to normalize the unit we got from MSE which is *millisecond²*. After RMSE, we get the value back into it's original unit millisecond.

3.8.3 R-Squared

R-Squared (RSQ) is one of the widely used and well accepted metrics for measuring significance of closure of data points of test set in terms of fitted line of regression. In both our MLR, DTR and RFR we used R-Squared to measure model accuracy in terms of predicting the user experience. The close the value is near to 1, the better accuracy the model has achieved to predict user experience. Let's say, for any actual line the UX is UX_a and for predicted line UX is UX_p . Let's say the mean value of UX is UX_m . For any i -th iteration in k -samples the equation of RSQ is as follows:

$$RSQ = 1 - \frac{\sum_{i=1}^k (UX_a - UX_p)^2}{\sum_{i=1}^k (UX_a - \bar{UX})^2} \quad (3.18)$$

3.8.4 Generalization

To emphasize equation 3.14 we demonstrate the generalization as follows:

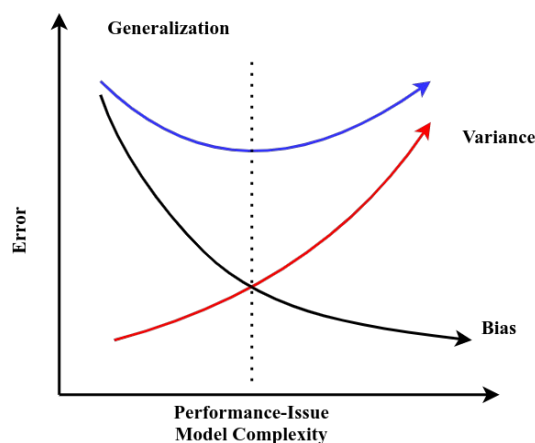


Figure 3.15: Tree Error Generalization

Our target in DTR and RFR is that we want to find a GE point which is where the error of predicting visual progression based user experience is lowest complex as possible. In our experiment of GE this happens as follows:

- From equation 3.14, When maximum depth of tree (MDT) gets increased, variance gets increased but bias gets decreased.
- When MDT gets decreased, variance gets decreased and bias gets increased.

3.9 API Development

To provide our research work as a service to the industries and developers, we decided to port our Machine Learning model to be deployed as a service. To do this, the final model we selected after feature selection, logic is being ported to statically typed language C# on top of .NET run time. Our API development process is:

- By using ML.NET a Machine Learning framework built on top of .NET framework, we ported the same Random Forest Regression tree considering the minimum sample leaf as well.
- After training ML.NET generates a ZIP file. This file contains all the trees, subsidiary decision trees with minimum sample leaf we defined during training. This ZIP file can be used by using File System provided by dotnet runtime from any application like console or Web API.
- To get request parameters, in our case, performance issues we developed our API endpoints to receive them from HTTP request body.
- We developed two endpoints as follows:
 - **Single Page Performance Decision:** This endpoint returns performance decision and the impact of improvement on visual progression in response.

- **Multiple Page Performance Decision:** This endpoint iterates over multiple page performance issues and returns decision after all iterations.

3.10 Deployment

Our final task with the entire research methodology ends with deployment which is where we first used the .NET Core 3.1 Runtime and built the service in available port. Then we created docker image, registered this to Heroku cloud by using Heroku container and finally pushed the entire docker image to complete the process.

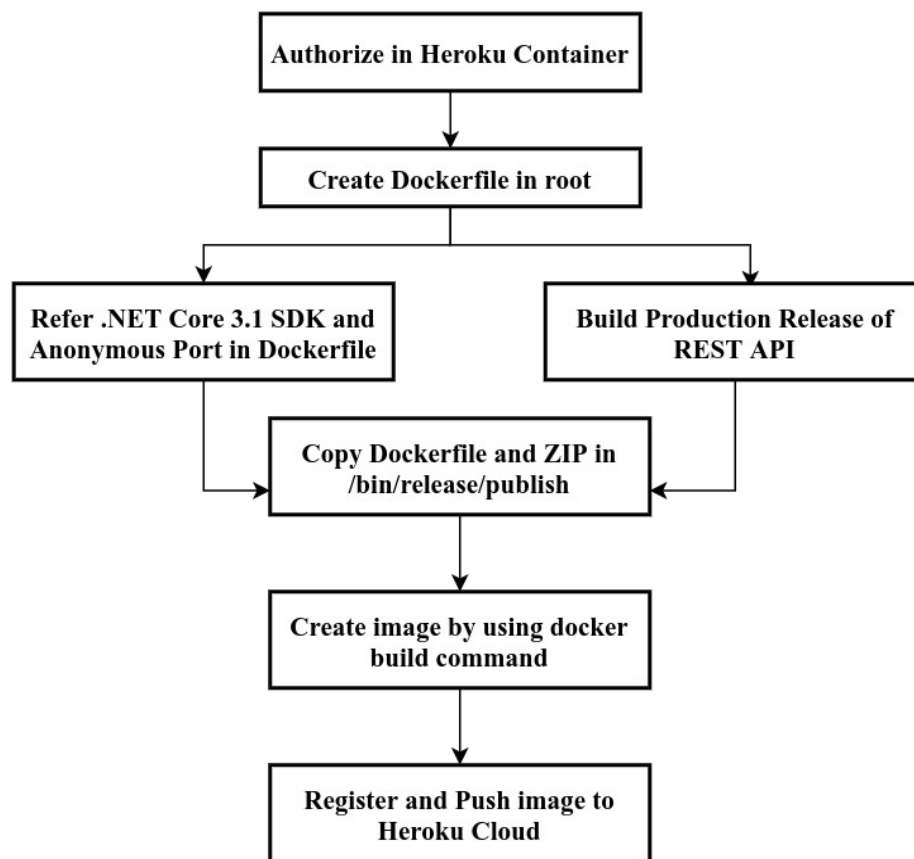


Figure 3.16: Model Deployment Process Using Docker

CHAPTER 4

RESULTS AND DISCUSSION

In this section, we will discuss the results of each experiment we have performed. This section demonstrates all the results starting from Univariate Feature Selection and Wrapper Method to deployment.

4.1 Univariate Feature Ranking

Our Univariate Feature Selection (UFS) experiment has been performed in two ways. Those UFS are F-Regression and Mutual Information Regression as follows:

Table 4.1: Univariate Feature Significance Ranking

F-Test Regression		Mutual Information Regression	
Rank	Significance	Rank	Significance
INTERACTIVE	72586.436	INTERACTIVE	0.534920
FCI	61568.783	FCI	0.502928
BUT	44338.991	TBW	0.398538
LCP	21309.618	BUT	0.380075
TBT	13574.026	LCP	0.299659
SRT	11948.695	FCP	0.285469
FMP	10205.972	FMP	0.251728
TBW	9722.405	TBT	0.249755
FCP	8979.385	SRT	0.194610
CLS	4760.827	CLS	0.126465
DOM_SIZE	548.658	DOM_SIZE	0.085601

From both of these ranking we found K-Best features by comparing their ranking as according to be interpreted as follows:

- INTERACTIVE and FCI are the most significant predictors of UX_INDEX.
- CLS and DOM_SIZE are the most insignificant predictors of UX_INDEX.

Based on these criteria, we eliminated CLS and DOM_SIZE for further Linear Regression algorithm based experiments.

4.2 Linear Regression Train-Test-Split Re-sampling

From the feature set we got 4.1, we have tried two re-sampling techniques, Train-Test-Split (TTS) and K-Fold Cross Validation (KCV). For k features from UFS, we have tried all possible $2^k - 1$ combinations in both re-sampling techniques which becomes 511 for 9 features by UFS. By using Scikit-Learn, a machine learning library built on top of Python programming language, we have tried to train most of our models. In the table above we demonstrate the results after 511 operations.

Table 4.2: Best Predictor Feature Set for Linear Regression with Train-Test-Split

Feature Set	R-Squared	MAE	RMSE
FCP, INTERACTIVE, SRT, BUT, FMP, TBT, LCP	0.7914	567.9890	820.6178
FCP, INTERACTIVE, SRT, BUT, FMP, TBT, FCI, LCP	0.7909	567.8335	821.6546
FCP, INTERACTIVE, SRT, BUT, FMP, TBT, TBW, LCP	0.7906	567.5409	822.22
FCP, INTERACTIVE, SRT, BUT, FMP, TBT, TBW, FCI, LCP	0.7903	567.4466	822.9401
FCP, INTERACTIVE, SRT, BUT, TBT, LCP	0.7894	572.6945	824.5147

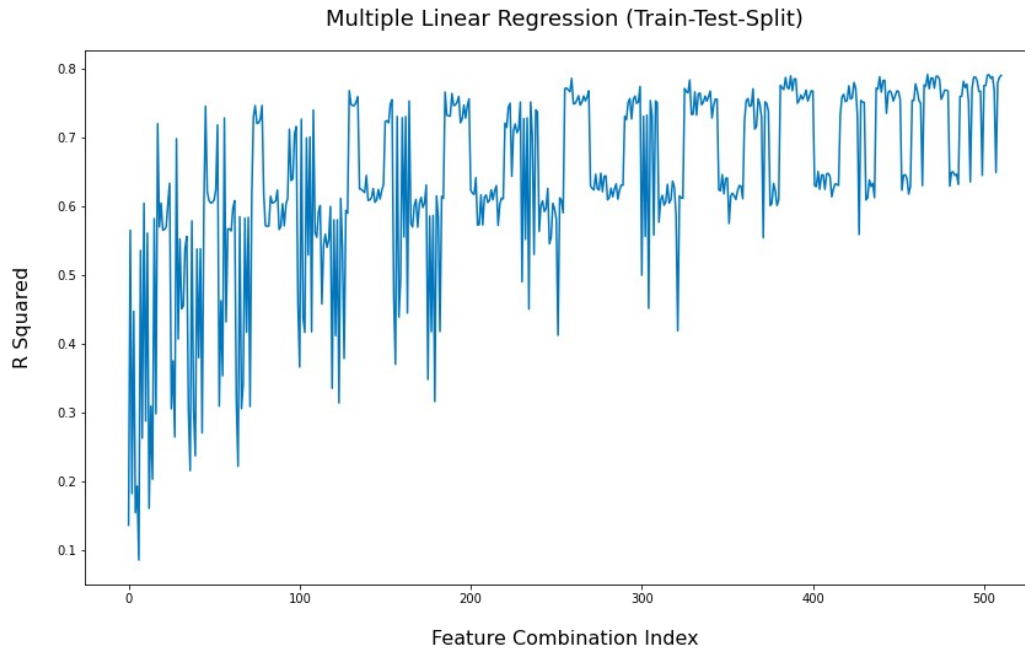


Figure 4.1: Linear Regression R-Squared On $2^k - 1$ Feature Combination

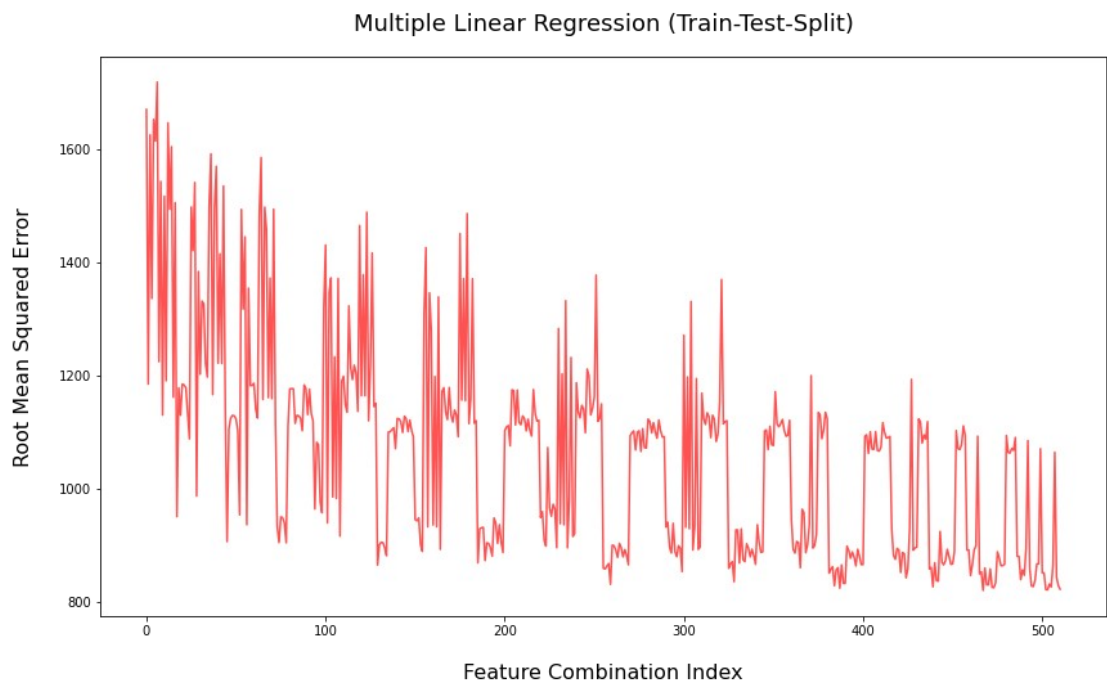


Figure 4.2: Linear Regression RMSE on $2^k - 1$ Feature Combination

Table 4.2 shows top 5 feature combination that have highest R-Squared value. In that table we can see that the feature combination UFS provided us captures the 4th best accuracy where by removing the feature FCI and TBW captures the highest accuracy. In terms of Root Mean Squared Error (RMSE), the first feature set has the lowest one. But in terms of Mean Absolute Error (MAE), the feature set with 9 features provided by UFS origin, has the lowest error recorded. On the other hand, Figure 4.1 demonstrates the variability of R-Squared in all possible 511 combinations and Figure 4.2 shows Root Mean Squared Error (RMSE) in all possible combinations. From both these two figures we can see that, the greater the feature presence is the greater the r-squared and the lower RMSE.

4.3 Linear Regression K-Fold Cross Validation

Linear Regression with TTS is great but the problem occurs in terms of consistency and robustness as earlier. So to keep consistency we have tried K-Fold Cross Validation in 3 segments for K = 5, 8 and 10. The results are demonstrated below:

Table 4.3: Linear Regression 3-Fold Cross Validation Summary

Fold	Feature Set	R-Squared	MAE	RMSE
3	FCP,INTERACTIVE,SRT,BUT, FMP,TBT,TBW,LCP	.7843	575.6362	23.9924
	FCP,INTERACTIVE,SRT,BUT, FMP,TBT,TBW,FCI,LCP	.7842	575.7532	23.9948
	FCP,INTERACTIVE,SRT,BUT, FMP,TBT,LCP	.7839	576.1556	24.0032

From Table 4.3 we can see that the first feature set eliminating FCI has the highest R-Squared and lowest RMSE. Preceding last feature set has lowest MAE recorded.

In another observation of 5 Fold Cross validation, we observed that the ranking of feature set is still the same but the R-Squared value comparing to 3-Fold KCV has decreased. But the interesting thing is that the MAE and RMSE has decreased as well which means the error rate has decreased.

Table 4.4: Linear Regression 5-Fold Cross Validation Summary

Fold	Feature Set	R-Squared	MAE	RMSE
5	FCP,INTERACTIVE,SRT,BUT, FMP,TBT,TBW,LCP	.7825	575.6315	23.9923
	FCP,INTERACTIVE,SRT,BUT, FMP,TBT,TBW,FCI,LCP	.7823	575.6624	23.9929
	FCP,INTERACTIVE,SRT,BUT, FMP,TBT,LCP	.7821	576.1370	24.0028

In terms of predicting visual progression based user experience we have finally tried 10 Fold Cross Validation in this segment. This still lacks in improving R-Squared but the interesting thing is MAE has increased for best case of feature set where RMSE is still the same. This leads us to a decision that the lower the fold is the greater the accuracy in terms of R-Squared.

Table 4.5: Linear Regression 5-Fold Cross Validation Summary

10	FCP,INTERACTIVE,SRT,BUT, FMP,TBT,TBW,LCP	.7807	574.1932	23.9623
	FCP,INTERACTIVE,SRT,BUT, FMP,TBT,TBW,FCI,LCP	.7806	574.2022	23.9625

The MAE and RMSE has variability in few cases. So, the best candidate in predicting UX_INDEX here is 3 Fold Cross Validation. If we compare, TTS with KCV results, we can conclude that R-Squared has decreased a little bit but there is significance in terms of RMSE getting reduces in KCV but MAE getting increased.

4.4 Wrapper Method Significance

At the beginning of Wrapper Method, we tried Backward Elimination. But the result was not well acceptable. Because, it demonstrates that all the feature are significant in terms of predicting user experience. So, we tried Recursive Feature Elimination (RFE) based on Random Forest Regression (RFR).

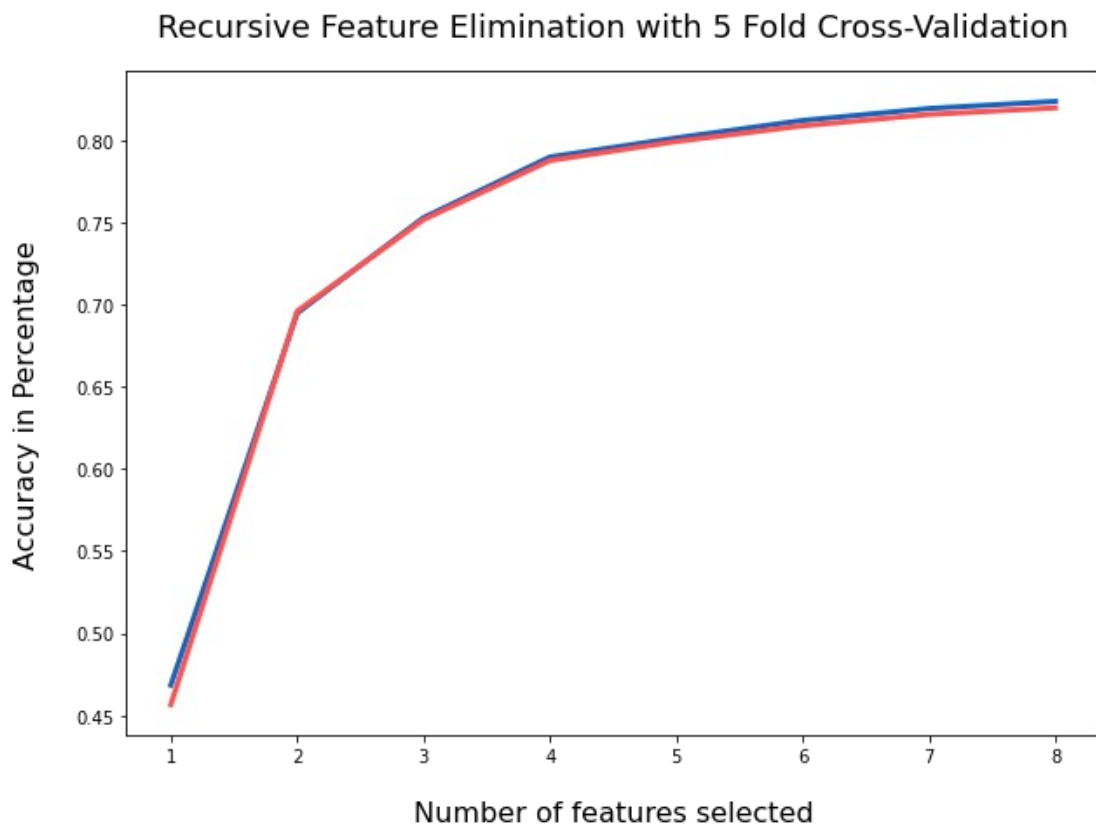


Figure 4.3: Recursive Elimination Impact of Features In Accuracy

In the figure above, Red line is the 3-Fold RFE and Blue line is the 5 Fold RFE. This plot demonstrates that the greater the feature number is the greater the accuracy will be. So, based on this Wrapper Method Decision, we tried tree based regression algorithms as well as considering all the features returned by UFS.

4.5 Decision Tree Regression Re-sampling

Before demonstration of Decision Tree (DT) results, we want to overview the feature sets by Univariate Feature Selection and Recursive Feature Elimination that leads best predicting the UX_INDEX as well as best accuracy.

4.5.1 Generalization

Our goal of doing Generalization was finding the best data point of minimum samples per leaf from training set and max depth of the DT. We used Root Mean Squared Error (RMSE) as the significant to normalize Bias-Variance trade off. Our observations are as follows:

Table 4.6: Lowest Generalization Error In Decision Tree Regression

Fold	Feature	Minimum Sample Per Leaf	Maximum Tree Depth	RMSE
8	RFE	10	4	1018.016777
		20		
10	UFS	10		
		20		

Our observations on reducing Generalization error shows that for both feature sets by RFE and UFS, RMSE remains still the same. So, we took maximum tree depth = 4.

4.5.2 8-Fold Cross Validation

From previous study, we got Maximum Tree Depth (MTD) = 4 as the best point for Bias-Variance trade off. Then we tried 8 Fold cross validation because, from previous study we got the result of both 8 and 10 fold are the same. So to reduce computation complexity, we took K = 8 as fold value and tried 8-Fold Cross validation:

Table 4.7: Best R-Squared and RMSE for RFE and UFS in Decision Tree

Feature	Minimum Samples Per Leaf	R-Squared	RMSE
RFE	40	0.7773	857.4690
UFS	50	0.7799	852.3275

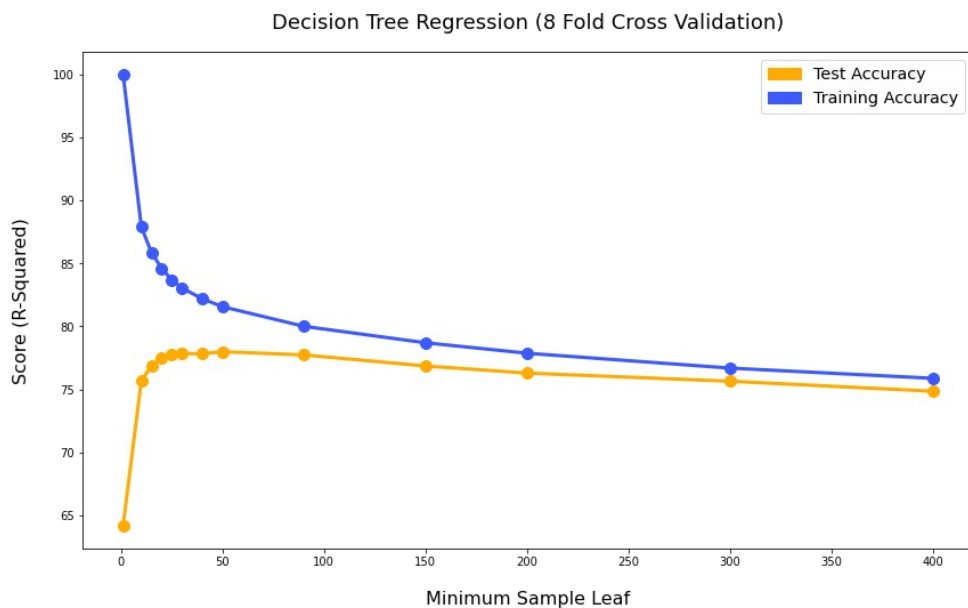


Figure 4.4: Decision Tree R-Squared for UFS Features

From Table 4.7, we got that the DT for UFS is more acceptable as demonstrated above to understand more in depth. Plot 4.4 shows that R-Squared is maximum till the

minimum samples per leaf is 50. After that both Training accuracy and Test Accuracy starts decreasing. Based on this segment we have another plot observation based on Root Mean Squared Error (RMSE).

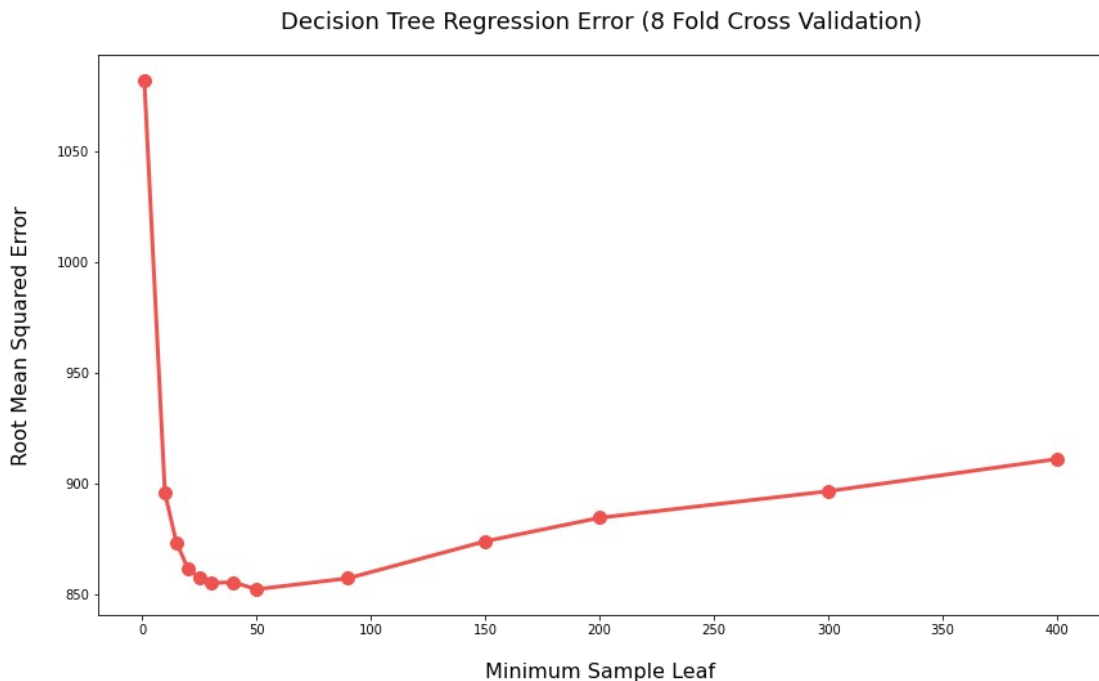


Figure 4.5: Root Mean Squared Error In Decision Tree Regression

Here in Figure 4.5, we can see that the RMSE is lowest as Minimum Sample Leaf = 50 and starts increasing onward.

4.6 Random Forest Regression Re-Sampling

We got to know about DT as the significant developed block for the algorithm Random Forest Regression (RFR) is one of the popular algorithm for capturing non-linearity in terms of predicting numerical response variable. So, in this segment we have first tested which K value is best for measuring visual progression value through RFR. Then we tried to find the best value of total number of decision trees also known as `n_estimators` in Scikit-Learn for which RFR is best fitted.

4.6.1 Best fold estimation

In the beginning of RFR, we tried different K-Fold Cross Validation to find the K best fold value that estimates the best R-Squared. From our observation, we found K = 8 to be the best fold value for RFR algorithm. This is being plotted as follows:

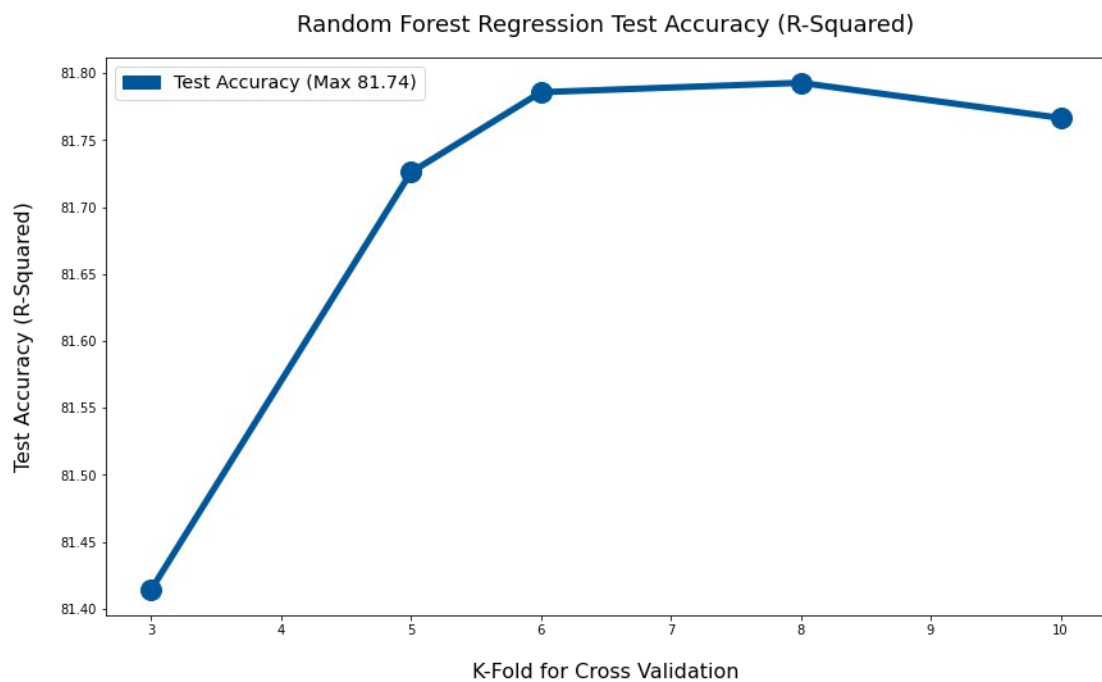


Figure 4.6: Best K-Fold Value in Random Forest Regression

From this, we found while K is 8, the tree estimates best visual progression based user experience without defining the number of estimators.

4.6.2 Best Total Estimators

While doing Random Forest Regression, we have to define number of estimators or, in our case, number of decision trees to strengthen the basic development block of the RFR for predicting visual progression. To do this, we tried estimating the score of the RFR with all possible estimators in between the range 10 to 200 with the gap of 10 in each iteration of estimation for predicting user experience.

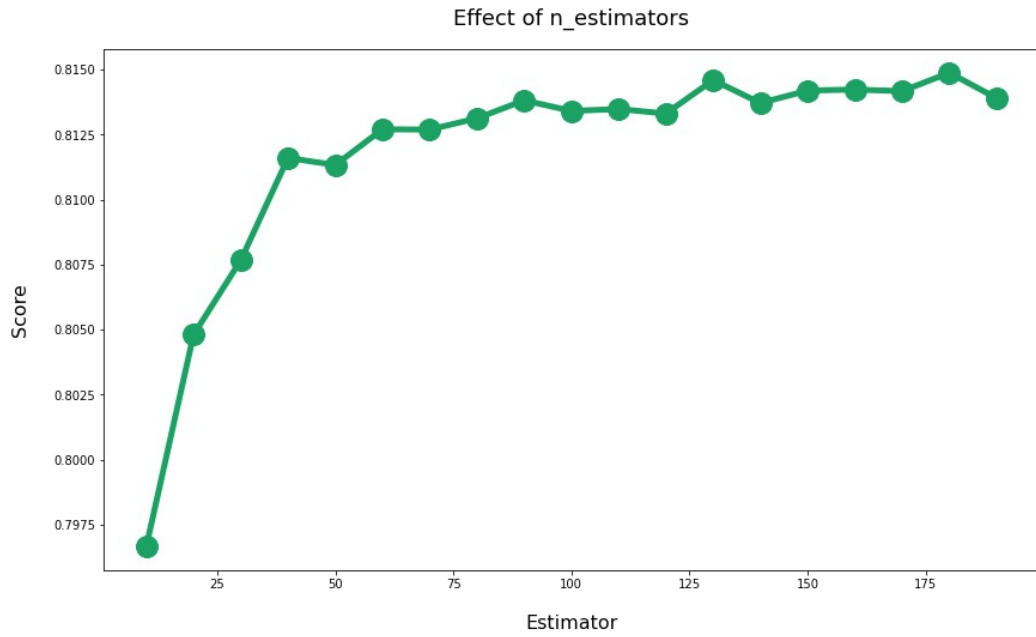


Figure 4.7: Estimator Effect on R-Squared with Random Forest Regression

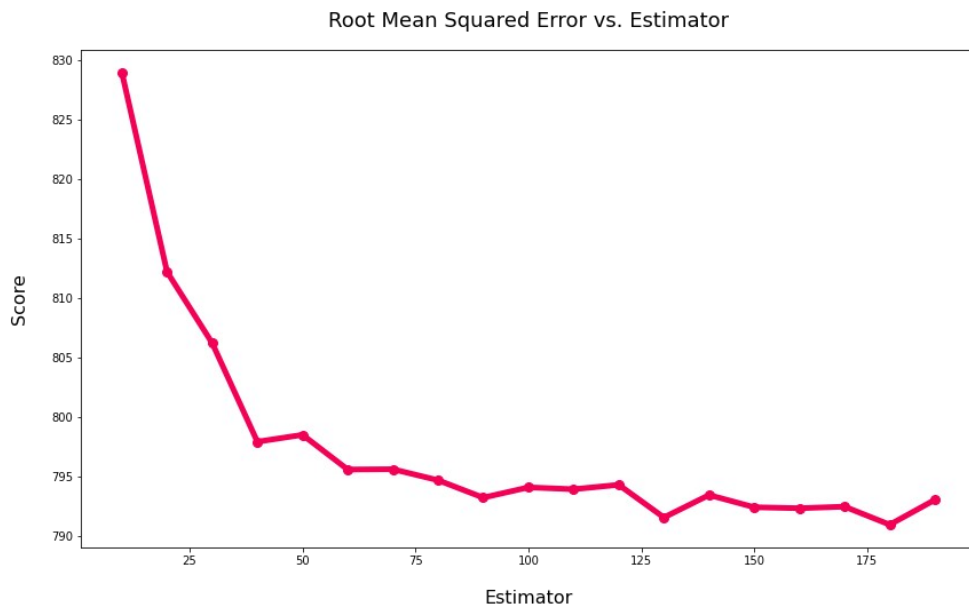


Figure 4.8: Estimator Effect on RMSE with Random Forest Regression

From both Figure 4.7 and 4.8, our observation is that for estimator value 180, RFR fits best in terms of predicting UX_INDEX based on web performance issues because at this point R-Squared is highest and RMSE is lowest considering all entries.

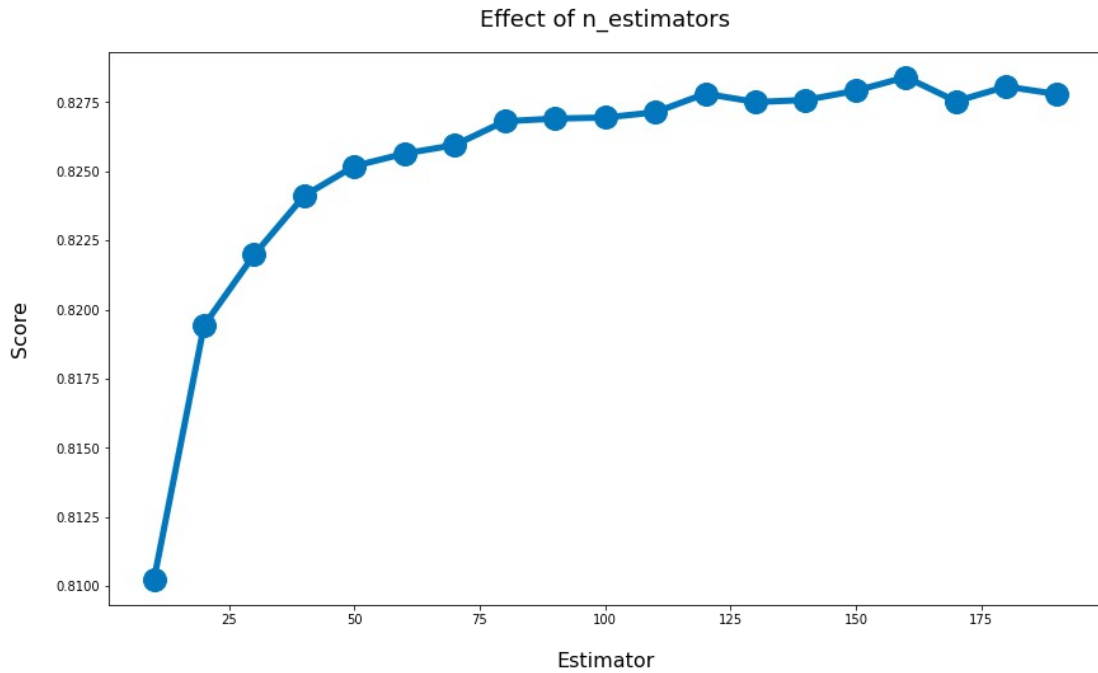


Figure 4.9: Best Estimator After Filtering in 1.5 times Interquartile Range

To justify our experiments in terms of finding best model, we did another experiment before final Random Forest Regression. We have tried removing UX_INDEX that are not in 1.5 times of Interquartile Range (25th Percentile of distribution of UX_INDEX to 75th Percentile of distribution). As we have discussed earlier, in our case, outliers of visual progression are just some sort of extreme values. After filtering in our dataset, Minimum UX_INDEX was 190 and Maximum was 7350.324 millisecond. After that, we have tried same procedure we have done so far in tree based experiments. We found following changes and similarities:

- 8 Fold cross validation still remains the best K value in Decision Tree and Random Forest Regression.
- Number of best estimator has decreased upto 20 from 180 to 160. From these insights we have tried experiment with new estimator value as well.

4.7 Decision Making

We compared all the experiment results we have performed so far in terms of R-Squared and RMSE. Here, MLR stands for Multiple Linear Regression, DTR stands for Decision Tree Regression and RFR stands for Random Forest Regression.

Table 4.8: Final Comparison Among Best Candidates

Re Sampling	Process	Feature Set	R-Squared	RMSE
Considering all entries				
TTS	MLR	FCP, INTERACTIVE, SRT, BUT, FMP, TBT, LCP	0.7914	820.6178
3 Fold	MLR	FCP, INTERACTIVE, SRT, BUT, FMP, TBT, TBW, LCP	0.7843	23.9924
5 Fold	MLR	FCP, INTERACTIVE, SRT, BUT, FMP, TBT, TBW, LCP	0.7825	23.9923
10 Fold	MLR	FCP, INTERACTIVE, SRT, BUT, FMP, TBT, TBW, LCP	0.7807	23.9623
8 Fold	DTR	FCP, INTERACTIVE, SRT, BUT, FMP, TBT, TBW, LCP, FCI	0.7799	852.3275
8 Fold	RFR	FCP, INTERACTIVE, SRT, BUT, FMP, TBT, TBW, LCP, FCI	0.8232	758.2418
UX_INDEX in Between (1.5 * Interquartile Range)				
8 Fold	DTR	FCP,INTERACTIVE,SRT, BUT,FMP,TBT,TBW,LCP,FCI	0.7930	707.1998
3 Fold	MLR	FCP, INTERACTIVE, SRT, BUT, FMP, TBT, TBW, LCP	0.7750	23.1100
8 Fold	RFR	FCP,INTERACTIVE,SRT, BUT,FMP,TBT,TBW,LCP,FCI	0.8265	646.2297
Tuning	RFR	FCP,INTERACTIVE,SRT, BUT,FMP,TBT,TBW,LCP,FCI	0.8336	632.9824

From above best candidate comparison table we observed following insights after running various experiments with different approaches and algorithms:

- Considering only R-Squared, Random Forest Regression gives the best result both considering all entries and even data points considering IQR.
- By observing 3 fold cross validation in Multiple Linear Regression we see that capturing all entries gives the better R-Squared comparing to considering IQR.
- If we focus only on RMSE, 3 fold cross validation in MLR considering both IQR and all entries are having lowest possible error rate.

Finally, after all observations we found RFR with tuning gives the best result in terms of RMSE. Based on the feature set we found feature importance in Figure 4.10.

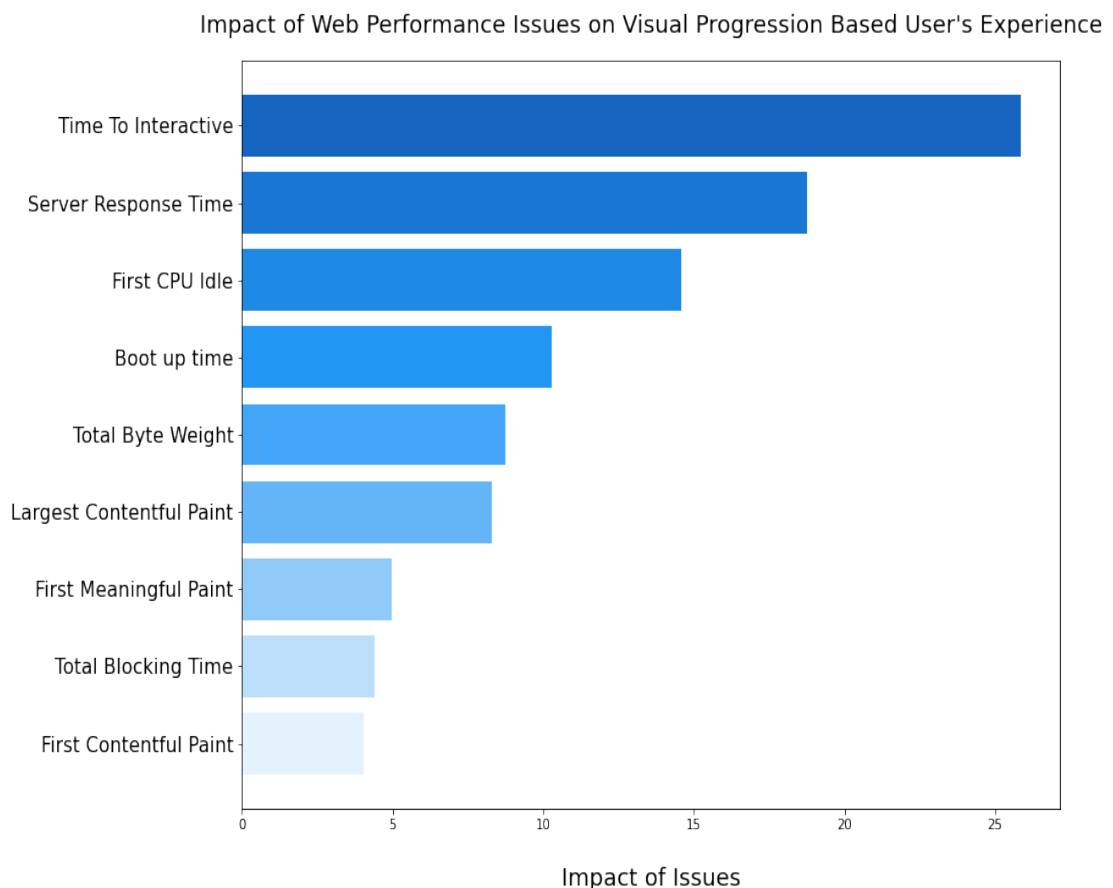


Figure 4.10: Feature Importance

4.8 Business Benefits of Conversion to PWA

From all our studies, we got to know about performance issues and user experience relation based on visual progression. To illustrate the research scope in terms of business and addressing the result of second research question, we collected the case studies from Karwatka (2018) et al. to a structured format data which includes 2 following things:

- Service Type (For example: E-Commerce, Food Ordering etc.)
- Increased User Conversion Rate After Conversion to Progressive Web Apps.

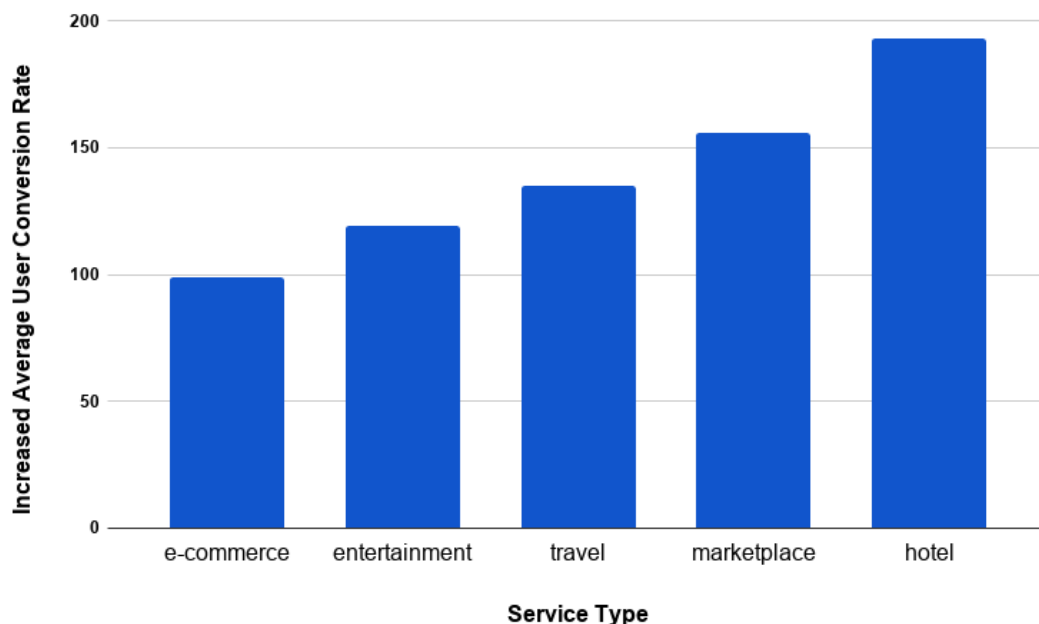
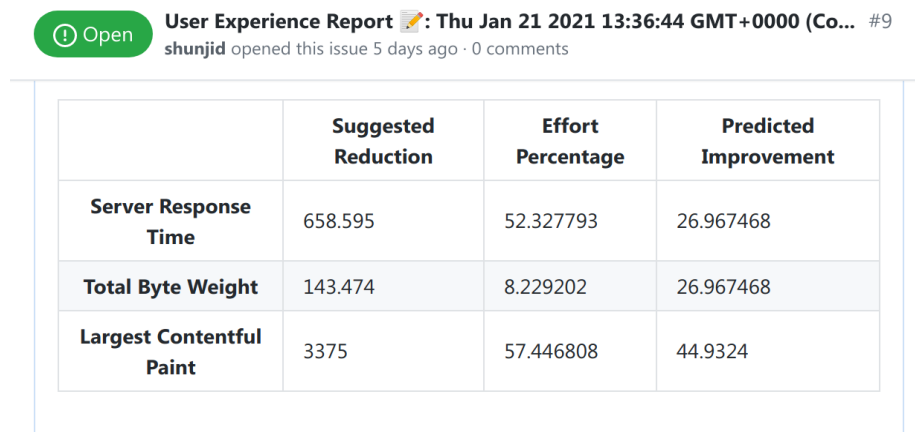


Figure 4.11: Average User Conversion Per Business Service Category

In Figure 4.11, We analyzed the Progressive Web Apps (PWA) case studies data and found that e-commerce sites that converted their site to a PWA are having average user conversion rate upto 98.88% where as hotel booking sites are having larger user conversion rate after performance improvement. So, from this analysis we can

conclude that spending time and budget on performance improvement and most likely for mobile user will return on investment as profit because of increasing user conversion rate based on smooth consumer experience on the web.

4.9 GitHub Actions Preview



The screenshot shows a GitHub Actions workflow preview for a job named 'User Experience Report'. The job is currently in a 'pending' state, indicated by a green circle with a downward arrow and the word 'Open'. The workflow was triggered by a push event on 'Thu Jan 21 2021 13:36:44 GMT+0000 (Co...)' and is identified as '#9'. The user 'shunjid' opened this issue 5 days ago, and there are 0 comments. Below the header is a table with three columns: 'Suggested Reduction', 'Effort Percentage', and 'Predicted Improvement'. The table contains three rows of data:

	Suggested Reduction	Effort Percentage	Predicted Improvement
Server Response Time	658.595	52.327793	26.967468
Total Byte Weight	143.474	8.229202	26.967468
Largest Contentful Paint	3375	57.446808	44.9324

Figure 4.12: GitHub Actions Preview

After developing the REST API by using dotnet and ML.NET, we developed a GitHub Actions Continuous Integration, Continuous Development (CI/CD) tool by using JavaScript running on Node.js runtime demonstrated above in Figure 4.12.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Findings

In our literature review, we found performance issues and various studies relevant to trending of user experience over time. We have tried different Feature Selection methods and tried the feature sets in various combinations and exchange with re-sampling techniques like Train-Test-Split and K-Fold Cross Validation with different algorithms to best predict the visual progression. After different set of experiments, we have found Random Forest Regression to have best R-Squared and Multiple Linear Regression having less RMSE. We finally developed a REST API to port the model in GitHub Actions to predict and recommend improvements in CI/CD logger.

5.2 Contributions

In this project based research development our contributions are listed as follows:

- Computationally Collect Data from Page Speed Insights REST API.
- Understanding Data through Exploratory Data Analysis.
- Based on Document Object Model visual progression, finding the impact of performance issues through feature significance.
- Development of REST API.
- Deployment using Docker Container to Heroku Cloud.
- Development of GitHub Actions.

5.3 Recommendation on future works

In this project based research, we have tried to find feature importance based on visual progression based user experience. Obviously, this is not the entire user experience. There are other user experience metrics in our dataset like Chrome CLS, LCP and FCP. To expand the research and project tasks, based on separate those experience goals, one can measure other importance of performance issues. For those who want to expand the project can elaborate it by designing an authorization mechanism to further analysis on privately available pages during recommendation.

REFERENCES

- Ramakrishnan, R., & Kaur, A. (2020). An empirical comparison of predictive models for web page performance. *Information and Software Technology, 123*, 106307. <https://doi.org/10.1016/j.infsof.2020.106307>
- AH, Z, M. (2018). *Analysis of web performance optimization and its impact on user experience* [Bachelor's dissertation, KTH Royal Institute of Technology]. KTH Royal Institute of Technology Research Repository. <https://kth.diva-portal.org/smash/get/diva2:1228341/FULLTEXT01.pdf>
- Trevisan, M., Drago, I., & Mellia, M. (2019). PAIN: A passive web performance indicator for ISPs. *Computer Networks, 149*, 115–126. <https://doi.org/10.1016/j.comnet.2018.11.024>
- Malavolta, I., Procaccianti, G., Noorland, P., & Vukmirovic, P. (2017). Assessing the impact of service workers on the energy efficiency of progressive web apps. *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, 35–45. <https://doi.org/10.1109/MOBILESoft.2017.7>
- Shivakumar, S. K. (2020). Web performance optimization framework. In S. K. Shivakumar (Ed.), *Modern Web Performance Optimization: Methods, Tools, and Patterns to Speed Up Digital Platforms* (pp. 49–78). Apress. https://doi.org/10.1007/978-1-4842-6528-4_3

Bocchi, E., De Cicco, L., & Rossi, D. (2016). Measuring the quality of experience of web users. *Proceedings of the 2016 Workshop on QoE-Based Analysis and Management of Data Communication Networks*, 37–42.
<https://doi.org/10.1145/2940136.2940138>

Hoßfeld, T., Metzger, F., & Rossi, D. (2018). Speed index: Relating the industrial standard for user perceived web performance to web qoe. *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, 1– 6.
<https://doi.org/10.1109/QoMEX.2018.8463430>

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python *Journal of Machine Learning Research*, 12, 2825–283

Jahromi, H. Z., Delaney, D. T., & Hines, A. (2020). Beyond first impressions: Estimating quality of experience for interactive web applications. *IEEE Access*, 8, 47741–47755. <https://doi.org/10.1109/ACCESS.2020.2979385>

Karwatka, P., & Wala, A. (2018). *Top 30 progressive web apps*. Top 30 Progressive Web Apps; Divante. <https://go.divante.com/top-30-progressive-web-apps/>

APPENDIX – A

List of Abbreviation

API	Application Programming Interface
DTR	Decision Tree Regression
HTTP	Hypertext Transfer Protocol
MLR	Multiple Linear Regression
MSE	Mean Squared Error
MAE	Mean Absolute Error
ML.NET	Machine Learning Dotnet
PWA	Progressive Web Apps
RMSE	Root Mean Squared Error
RFR	Random Forest Regression
REST	Representational State Transfer
RFE	Recursive Feature Elimination
UFS	Univariate Feature Selection

APPENDIX – B

Plagiarism Report

2/1/2021

Turnitin

Document Viewer

Turnitin Originality Report

Processed on: 01-Feb-2021 13:16 +06
ID: 1498887615
Word Count: 12891
Submitted: 1

171-35-1862 By Shunjid Rahman Showrov

Similarity Index	Similarity by Source
3%	Internet Sources: 2% Publications: 1% Student Papers: 2%

[include quoted](#) [include bibliography](#) [excluding matches < 10 words](#) mode:
quickview (classic) report [print](#) [refresh](#) [download](#)

<1% match (Internet from 15-Mar-2020) http://dspace.daffodilvarsity.edu.bd:8080	<input type="checkbox"/>
<1% match (Internet from 23-Aug-2020) http://dspace.daffodilvarsity.edu.bd:8080	<input type="checkbox"/>
<1% match (Internet from 01-Apr-2020) https://www.slideshare.net/RaihanMahmud5/remote-doctor-project-report	<input type="checkbox"/>
<1% match (student papers from 30-Oct-2017) Submitted to United International University on 2017-10-30	<input type="checkbox"/>
<1% match (Internet from 28-Jul-2019) http://dspace.daffodilvarsity.edu.bd:8080	<input type="checkbox"/>
<1% match () https://scholars.unh.edu/thesis/621	<input type="checkbox"/>
<1% match (Internet from 17-Mar-2020) http://www.wis.wtn.tue.nl	<input type="checkbox"/>
<1% match (Internet from 07-Aug-2017) http://ulspace.ul.ac.za	<input type="checkbox"/>
<1% match (student papers from 20-May-2010) Submitted to Cavendish College on 2010-05-20	<input type="checkbox"/>
<1% match (Internet from 10-Jan-2021) https://www.coursehero.com/file/67301064/Lab-Assignment-1pdf/	<input type="checkbox"/>
<1% match (publications)	

https://www.turnitin.com/newreport_classic.asp?lang=en_us&id=1498887615&ft=1&bypass_cv=1

1/22