


RightPrice – AI Based Price Estimation System

A Project Report
Presented to
The Faculty of the Computer Engineering Department
San Jose State University
In Partial Fulfillment
Of the Requirements for the Degree
Bachelor of Science in Software Engineering
Bachelor of Science in Computer Engineering

By
Manmeet Gill, Sidarth Shahri, Manpreet Singh, Karthik Tella
12/2019

Copyright © 2019
Manmeet Gill, Sidarth Shahri, Manpreet Singh, Karthik Tella
ALL RIGHTS RESERVED

APPROVED FOR THE COLLEGE OF ENGINEERING



Dr. Vishnu Pendyala, Project Advisor

Professor Rod Fatoohi, Instructor

Dr. Xiao Su, Computer Engineering Department Chair

ABSTRACT

RightPrice – AI Based Price Estimation System

By Manmeet Gill, Sidarth Shahri, Manpreet Singh, Karthik Tella

Online shopping has grown massively over the past couple of years. For example, Amazon is one of the biggest websites for online shopping. Another very popular online market is buying/selling old and used items. Instead of having an old device lying around, it makes sense to want to get some money for it instead. However, there is no real way of knowing the prices of those items. It would not make sense to overpay for an older device. In the same sense, it would also not make sense to sell a valuable device for a very cheap price.

Manually visiting several seller websites and looking at various listings is not feasible if one wished to unload many products in a short time period. Looking for price quotes on the market for a variety of products is very time-consuming. Therefore, there is pretty much no effective way to find the average price of an item. Additionally, there are issues with sellers misquoting their prices, both accidentally and intentionally. Additionally, the price of used products on the market should degrade with time as the product deteriorates, occupies valuable virtual “shelf space”, and continues to be a burden on the seller. This follows that the seller would want to lower the price to get rid of the item faster or to make the sale.

Our product would essentially do most of the tedious work for the user, and return a value that is the average price of the given product based on the data gathered by our

product. This would be possible through web scraping data from various used marketplaces such as Amazon, Craigslist, Ebay, etc. Then, using and filtering the dataset obtained from web scraping, AI models can be trained to predict price of certain item given some parameters based on average price of similar items in the dataset. For our project, we focused on a single category of used goods: smartphones. We chose this category as it was the most popular used goods sold and bought on online marketplaces.

Acknowledgments

This project would not have been possible without the help of Dr. Vishnu Pendyala. His guidance helped us get around many blockers and pushed us towards success.

Professor Rod Fatoohi also helped in many ways, especially in regard to realizing the sociological impacts our project may have if released to the public.

Table of Contents

Chapter 1 Introduction

- 1.1 Project Goals and Objectives
- 1.2 Problem and Motivation
- 1.3 Project Application and Impact
- 1.4 Project Results and Deliverables
- 1.5 Project Report Structure

Chapter 2 Background and Related Work

- 2.1 Background and Used Technologies
- 2.2 Literature Survey
- 2.3 State-of-the-art Summary

Chapter 3 Project Requirements

- 3.1 Domain and Business Requirements
- 3.2 System (or Component) Functional Requirements
- 3.3 Non-functional Requirements
- 3.4 Context and Interface Requirements
- 3.5 Technology and Resource Requirements

Chapter 4 System Design

- 4.1 Architecture Design
- 4.2 Interface and Component Design
- 4.3 Structure and Logic Design
- 4.4 Design Constraints, Problems, Trade-offs, and Solutions

Chapter 5 System Implementation

- 5.1 Implementation Overview
- 5.2 Implementation of Developed Solutions
- 5.3. Implementation Problems, Challenges, and Lessons Learned

Chapter 6 Tools and Standards

- 6.1. Tools Used
- 6.2. Standards

Chapter 7 Testing and Experiment

- 7.1 Testing and Experiment Scope
- 7.2 Testing and Experiment Approach
- 7.3 Testing and Experiment Results and Analysis

Chapter 8 Conclusion and Future Work

References

Appendix: Data on Smartphone Prices

List of Figures

Chapter 1 Introduction

Figure 1A. Front-end Implementation of the Search Feature	Page 14
Figure 1B. Front-end Implementation of Results	Page 14
Figure 1C. Example API Response	Page 15

Chapter 3 Project Requirements

Figure 3A. Process Summary Diagram	Page 24
Figure 3B. Class Diagram for our client application	Page 25
Figure 3C. Class Diagram for our web scraper	Page 25
Figure 3D. Activity Diagram for RightPrice	Page 26

Chapter 4 System Design

Figure 4A. Interface and Component Diagram	Page 31
Figure 4B. Structure and Logic Design	Page 32
Figure 4C. User Access and Application Flow	Page 33

Chapter 5 System Implementation

Figure 5A. Code snippet showing locations where data was scraped	Page 40
Figure 5B. Statistics of Collected Data Points	Page 40
Figure 5C. Code Snippet of Data Cleaning	Page 41
Figure 5D. Code snippet showing data preprocessing using OneHotEncoder	Page 42
Figure 5E. Figure showing results received when API is called	Page 44
Figure 5F. User Entry Form	Page 45
Figure 5G. User Results Page	Page 45

Chapter 7 Testing and Experiment

Figure 7A. Test results for Training and Testing Data	Page 61
---	---------

Appendix: Data on Smartphone Prices

Figure AA. Price vs. Quality on a Scale from 1 to 5	Page 67
Figure AB. Average prices of devices based on manufacturer	Page 68

List of Tables

Chapter 1 Introduction

Table 1A. Short Excerpt of Collected Data	Page 16
Table 2A. Courses relevant to project taken at SJSU	Page 20

Chapter 2 Background and Related Work

Table 2A. Course relevant to project taken at SJSU	Page 19
Table 2B. Course relevant to project taken at Udemy.com	Page 20

Chapter 3 Project Requirements

Table 3A. Functional Requirements	Page 26
Table 3B. Non-functional Requirements	Page 27
Table 3C. Technologies and Resources	Page 29

Chapter 4 System Design

Table 4A. Regular categorical non-numeric data in the original set	Page 37
Table 4B. Numeric data preprocessed using OneHotEncoder	Page 37

Chapter 5 System Implementation

Table 5A. Table showing a few rows of collected and cleaned data	Page 42
--	---------

Chapter 7 Testing and Experiment

Table 7A. Scope of the tests and expected results before testing	Page 58
Table 7B. Actual results after testing	Page 60

Chapter 1. Introduction

1.1 Project Goals and Objectives

The project's main goal was to empower customers who are buying and selling items on online marketplaces for used goods such as Craigslist, Facebook Marketplace and eBay. Our product aimed to utilize advertisements from online marketplaces as precepts to train a learning agent to utilize the knowledge gained from these precepts to provide estimates and analytics about the availability and price of desired products. Thus, our secondary goals were to extract price and other data from an online marketplace using a web scraping algorithm, to build a machine learning model using state-of-the-art tools and technology, and to present the user with a tool to search for the price of an item. These secondary goals would help us accomplish our overall main goal of empowering customers who were looking to buy or sell used goods. Our project had the potential to vastly improve the online marketplace experience by giving customers the knowledge necessary to inform their purchases. As a small addendum, we decided to focus on a single category of used goods to simplify the data collection and machine learning model due to the scope and timeline of the project. We chose to focus on smartphones.

1.2 Problem and Motivation

Our project was born out of personal experience that resonated with other peers. That is, shopping or selling used goods was a problematic experience as it was not clear

how much a used good should cost. In the case of smartphones, several factors determine the market price of the item: age, condition, carrier, brand, model, storage capacity, technical specifications, etc. Thus, we saw a problem and devised our project as a potential solution. Our project would learn the market trends for specific used products based on data from online marketplaces and give users that information to better inform their decisions.

The main problems with our project were mostly technical and legal. Technical problems included making sense of online advertisements, which were mostly unstructured, web scraping from dynamic websites, and hosting our learning agent. Legal problems mainly came down to the accessing rights on websites. Some websites were against third parties accessing their source code and benefiting from it, thus claiming them to be property. Such websites can be scraped but usually should not be scraped without the consent of the domain owner. Therefore, gaining rights or consent to access some marketplaces was a problem. The main motivation of the project was to empower online customers and sellers given the vast and growing marketplaces online.

The needs for our project mainly included ad data, hosting space and source code/data access privileges to preexisting marketplaces. With online marketplaces for used products gaining popularity, the sheer number of advertisements could overwhelm buyers and sellers in making a decision. Our project used web scraping and machine

learning algorithms to analyze a big portion of these ad space to analyze common trends for a given product simplifying the process of researching and decision making for users.

1.3 Project Application and Impact

The goal of our project was to be able to return to the user an accurate price of an item on the used marketplace. Our completed project has the potential to directly affect the used marketplace. Our project also has the potential to impact academics, society, and industry. Firstly, in college, students tend to buy used textbooks because they tend to be a lot cheaper than buying brand new. Our project helps with this process because students are now able to get a general price for the textbook, and they now have the power to find the best deal. Students are now able to find expensive materials such as laptops, textbooks, and software for a more accurate market price with our project. This also applies to society. Technology is becoming more and more advanced, and with it, prices have also been gradually going up. For example, the iPhone 11 Pro costs around \$1000 for a brand new device. This is where the used marketplace comes in handy. Consumers can buy slightly used technologies for a fraction of the price, making some of these expensive technologies obtainable for the lower/middle class. This will also obviously have an impact on industry as used marketplaces take customers away from the new marketplaces. However, the smartphone industry will not take a direct hit in terms of revenue because in order for an item to be on the used marketplace, someone has to have

bought it first. In the future, if the used marketplace becomes very popular, it could take a large number of customers away from buying new items.

1.4 Project Results and Deliverables

The deliverables for this project were

- Front-end application
- Back-end application/API
- Machine learning model
- Script for data collection
- Project report

Through this project, we presented a scalable and highly available full-stack application where the user is able to search for an item and receive an accurate price for that item. We built a clean and easily usable front-end application where users can use a GUI tool to get a price prediction for an item they are searching for. The front-end application was implemented using ReactJS.

RightPrice GETTING STARTED SEARCH RESULTS ABOUT THE TEAM

Search for an item:

Phone type: Apple

Color: Space Grey

Condition: Good

Contract: Unlocked

Memory: 64GB

Mobos: Apple iOS

Model: iPhone 8 Plus

SEARCH

Figure 1A. Front-End Implementation of the Search Feature

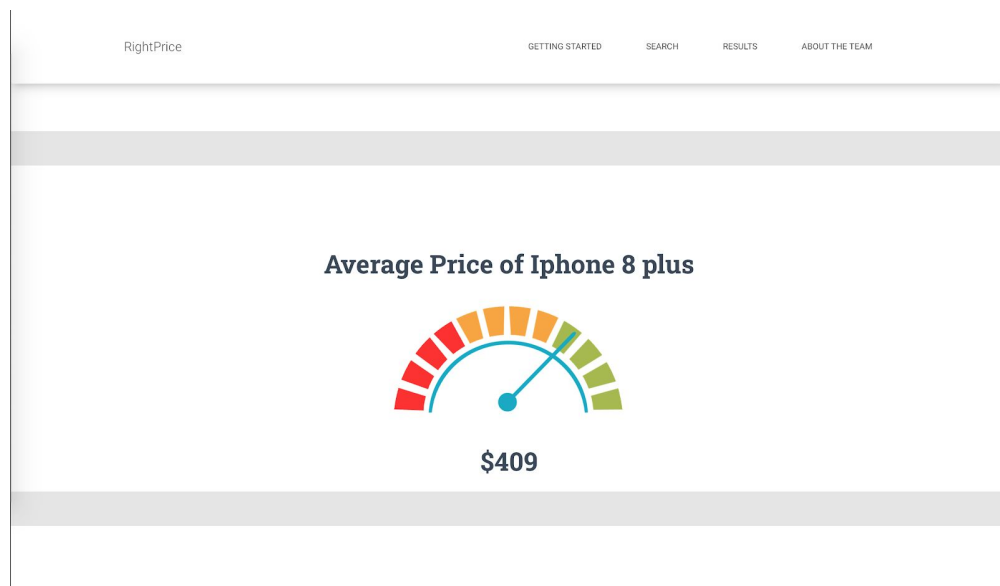


Figure 1B. Front-End Implementation of Results

We also designed and implemented a back-end web service/API. There were four parts to the back-end: data collection, data preprocessing, the machine learning model,

and the web service/API. Our backend was hosted on an AWS cloud provider and was accessible as a public web URL or an API call. We also obtained data from various used marketplaces, and cleaned that data to be able to calculate an accurate average price of a given item using a machine learning algorithm. For our machine learning model, we wrote an algorithm solution using a KNN (K-Nearest Neighbors) algorithm to provide an accurate prediction on the price of a requested item. We also wrote a report detailing our efforts in researching our topic, designing and implementing our project, and testing and examining the results.

```
{  
  "predict": "402.0",  
  "nn": "[500. 570. 250. 575. 115.]"  
}
```

Figure 1C. Example API Response

Table 1A. Short Excerpt of Collected Data

BRAND	COLOUR	CONDITION	CONTRACT	MEMORY	MOBOS	MODEL	PRICE
Samsung	gray	like new	Sprint	256	android	S8 PLUS	\$270.00
Blu	red	like new	Unlocked	16	android	Y85	\$50.00
Samsung	Black	Good	Verizon	32	android	S10 plus	\$745.00
Apple	gray	like new	T-Mobile	64	apple iOS	iPhone 8 Plus	\$400.00
Samsung	white	excellent	Unlocked	64	android	NOTE 8	\$70.00
Apple	white	like new	Unlocked	256	apple iOS	XR	\$400.00
Apple	white	like new	Unlocked	64	apple iOS	6 PLUS	\$220.00
Apple	black	good	Unlocked	32	apple iOS	6S	\$150.00
Apple	rose	good	Unlocked	32	apple iOS	6	\$110.00
Apple	black	excellent	Unlocked	32	apple iOS	6 PLUS	\$155.00
Apple	rose	excellent	At&t	64	apple iOS	6	\$140.00
Apple	black	good	Verizon	64	apple iOS	6	\$135.00
Apple	silver	excellent	At&t	128	apple iOS	IPHONE 7	\$225.00
Apple	white	Good	Unlocked	256	apple iOS	XS	\$600.00
Apple	white	like new	Unlocked	128	apple iOS	IPHONE 7 PLUS	\$275.00
Apple	white	like new	Unlocked	64	apple iOS	iphone 8	\$365.00
Samsung	gray	excellent	T-Mobile	128	android	S10 PLUS	\$180.00
Apple	white	excellent	At&t	128	apple iOS	IPHONE 7 PLUS	\$400.00
Samsung	black	excellent	Unlocked	64	android	S10	\$199.00

1.5 Project Report Structure

This following sections of this project report contain more on how we designed and implemented our project, the tools we used, and the testing process we used to uncover issues in our design. Other sections in the report include research we found on our topic of choice as well as the result of our project after implementation.

In more detail, Chapter 1 contains a brief introduction of our project, including the project's goals, context, motivation, and the expected results and deliverables. Chapter 2 contains information on the background of the project. That is, our knowledge of the technologies we would need to implement the project and any research we did on our topic. It also contains a section of the state-of-the-art technologies we used. Chapter 3 contains information on our project requirements. It describes our system and functional requirements, non-functional requirements, context and information requirements, and technology and resource requirements. Chapter 4 contains information on the design of our implementation of the project. That is, the architecture design, the interface and component design, the structure and logic design, the design constraints and challenges, and the solutions or trade-offs we came up with for those constraints and challenges. Chapter 5 contains information on the actual implementation of our project including the implementation of our back end, front end, and data collection algorithms. It also details any challenges overcome and lessons learned from the implementation. Chapter 6 details the tools and standards used in our project. Chapter 7 details the testing environment of our project and results and analysis of those results. Finally, Chapter 8 details our conclusion and any changes we would make should we continue to improve on this project in the future. Several figures and tables throughout the report help explain our design and our results. They are labeled with the number of the chapter they belong in alongside a letter indicating the order of appearance.

Chapter 2 Background and Related Work

2.1 Background and Used Technologies

The project was divided into three main segments: data procurement, data analysis (learning), and interface design. Data procurement mainly dealt with accessing advertisements from online marketplaces. To make the process more seamless our project implemented web scraping using Python 3 with the Scrapy library. From the Scrapy library, we used a tool called Spider to collect our data. As a part of procurement, we structured the data and saved it in CSV (.csv or comma separated values format). We heavily used Python 3 and Microsoft Excel with the help of Pandas library to clean our data (removing inaccurate postings).

Data analysis mainly encompassed our intelligent agent which took the data from the previous step and trained a model written using Python 3 and tools in the Scikit-learn library. It was built using the K-Nearest Neighbors algorithm.

The interface of the project mainly dealt with our GUI and hosting. The GUI was developed using JavaScript with ReactJS as our choice of library. Some of the other dependencies of GUI included npm, webpack and other third-party styling libraries. Hosting and backend was mainly done on AWS with technologies such as dynamoDb and Django being used to write the application.

Table 2A. Courses relevant to project taken at SJSU

Course	Course Title	How It Was Relevant
CMPE 188	Machine Learning and Big Data	This course was taught by our very own advisor. Using information learned from this class, we were able to build a machine learning model using the KNN model.
CMPE 174	Server Side Application Development	This course offered information on how to develop an application on a server. This was extremely useful for our project as our application required a back-end to make API calls to and retrieve data from.
CMPE 131	Software Engineering I	This course offered knowledge on proper engineering practices and the software development cycle. We learned how to use the Agile methodology in this course.
CMPE 133	Software Engineering II	This course further expanded on the software engineering process. Knowledge in this class helped us properly develop software application for our project.
CMPE 172	Enterprise Software Development	This course taught our team members about the state of the art technology used in enterprise software development. For example, we used AWS-EC2 to deploy our back-end application.
CMPE 187	Project Development and Management	This course detailed how to develop a software project and how to manage a team. This was especially helpful when trying to organize the various components

		of the project and our team members.
CMPE 165	Advanced Algorithm Design	This course taught us how to develop algorithms for various uses. The knowledge in this course helped us to develop our web scraping algorithm.
CMPE 126/CS 146	Data Structures and Algorithms	This course offered a fundamental tutorial on basic data structures and algorithms and was a starting point for the entire project.
MATH 161/ISE 130	Statistics	Measuring the performance of our application, the accuracy of our machine learning model, and analyzing the data from our application required knowledge from this statistics course.
ISE 164	Human Computer Interaction	This course helped us with user interface design and prototyping.

Table 2B. Courses relevant to project taken at Udemy.com

Course Title	How it was relevant
Making Django Applications	Since our project required a back-end application, we needed to take additional coursework to learn how to build an application with Python. This Django course proved to be the perfect fit for us. We used knowledge from this course to build the API and back-end service for our project.
A Complete React Guide	Building a user-friendly and attractive front-end application was a must for us. Thus, we elected to use React to design the application. The knowledge from this online Udemy course was particularly helpful for this.

2.2 Literature Search

This section details some background research we did on the topic of buying and selling used goods. The research in this section also covers some of the state-of-the-art technology used in the implementation of the project.

According to a Deloitte Global research paper, the used smartphone market is one of the fastest growing markets which has an economic value of about 17 billion with an annual increase of about 80 million every year (Deloitte, 2017, p. 1) [2]. Our application provides an estimate on prices of the smartphone given the information about physical and internal attributes of the smartphone using machine learning algorithms on data from other advertisements on public marketplaces. Any product providing similar service for same domain i.e. smartphones is not yet available in the market. Dr. Zafar Khan and Dr. Asim, from University of Engineering and Technology, Lahore, has written a research paper which revolves around using classification to predict prices of new smartphones using dataset available on GSMarena.com (Asim, 2018, p. 2) [1]. Our solution solved the problem of underpaying or overpaying for a used product.

Machine learning is a new field of technology that can be used for the purpose of estimation and predictions. These predictions depend heavily on the data given to the algorithm. As Poursaeed states in his article, Zillow has a home estimation algorithm powered by machine learning. It is called “Zestimate” and is calculated using a

proprietary formula for over 100 million homes in the United States (Poursaeed, 2018, p.1) [7]. Its median error is about 8% (Poursaeed, 2018, p. 1) [7]. Redfin is another major real estate company that has a similar estimation algorithm. This goes to show how important machine learning can be in terms of predicting the “cost” of something. In our project we aim to accurately predict the price of used smartphones, which is somewhat similar to what Zillow and Redfin have done. Obviously our project scope is completely different, but the idea of cost estimation has been implemented before using machine learning algorithms.

Scikit is a fantastic library that offers important tools in the use of machine learning algorithms, according to Fabian in his article “Scikit-learn: Machine learning in Python”(Fabian, 2011, p. 1) [3]. His article outlines the kind of problems that can be solved with scikit. For example, in a typical machine learning problem, a developer considers a set of n samples of data and tries to predict properties of not-yet-recorded or unknown data. Training a piece of software to learn from given data to predict these properties falls under two domains: supervised and unsupervised learning. In supervised learning, the problem is either a classification problem or a regression problem. In classification, data is classified into two or more classes and the model is required to classify future or unknown data. In regression, the desired output of the model is to follow the parameters of the input data.

These research references helped influence our decisions in which state-of-the-art technologies to use in the implementation of our project.

2.3 State-of-the-art Summary

The used marketplace is a large portion of where people buy things. It simply just makes sense to save a bit of money and get a product that has been used before. The only downfall, however, is overpaying for a product. The same can be said for selling. As of right now, there are no technologies that look through all of these popular used marketplaces and give a “good” price for an item. The technologies we used helped ease the development process. They were very reliable and were the best methods at the time of development. For web scraping we used Python scripts and the Scrapy library. For cleaning and preprocessing of the collected data, we used OneHotEncoder from the Scikit-learn library. For the machine learning model, we used KNN (K-Nearest Neighbors) regressor algorithm from Scikit-learn. For the development of the back-end application, we used Python and Django. For the development of the front-end application, we used ReactJS. We used cloud solutions provided by AWS for the deployment of our applications.

Chapter 3 Project Requirements

3.1 Domain and Business Requirements

Below are diagrams showing the user flow and data retrieved for our application.

Figure 1 shows a summary of the process. Figure 2 shows a class diagram and how data is stored in the front-end application for our project.

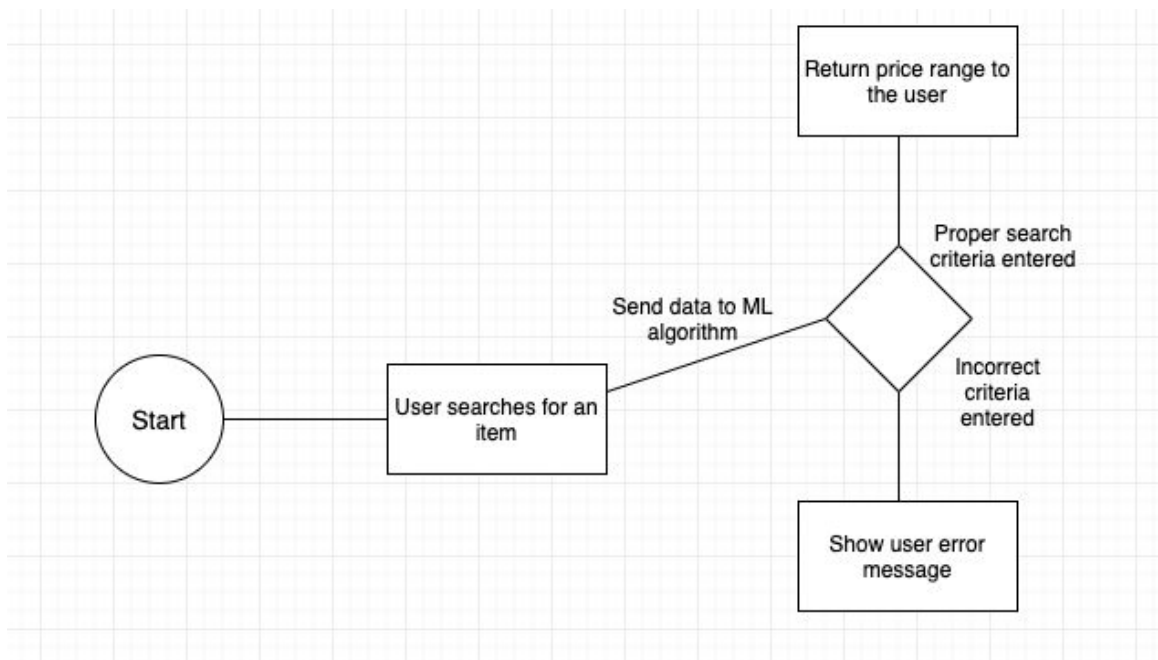


Figure 3A: Process Summary Diagram

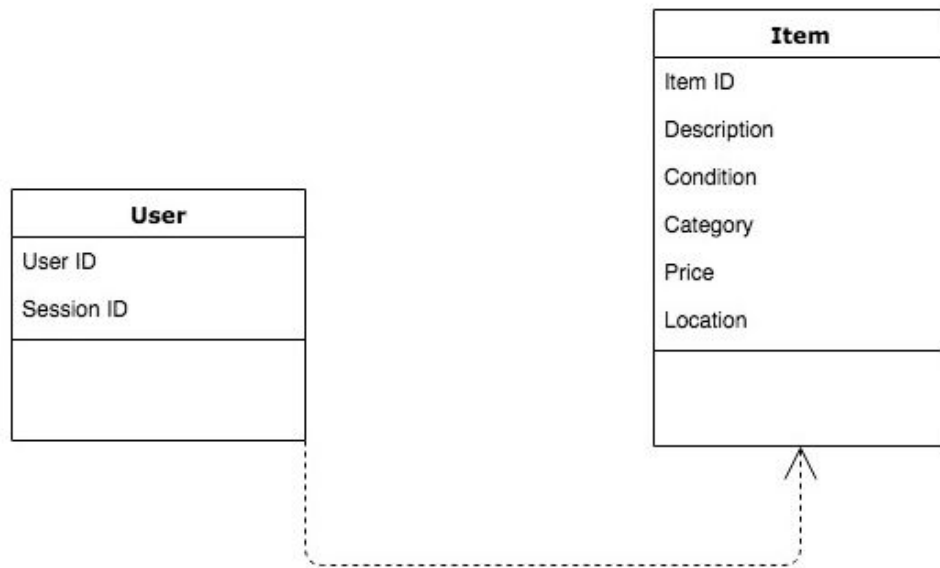


Figure 3B: Class Diagram for our client application

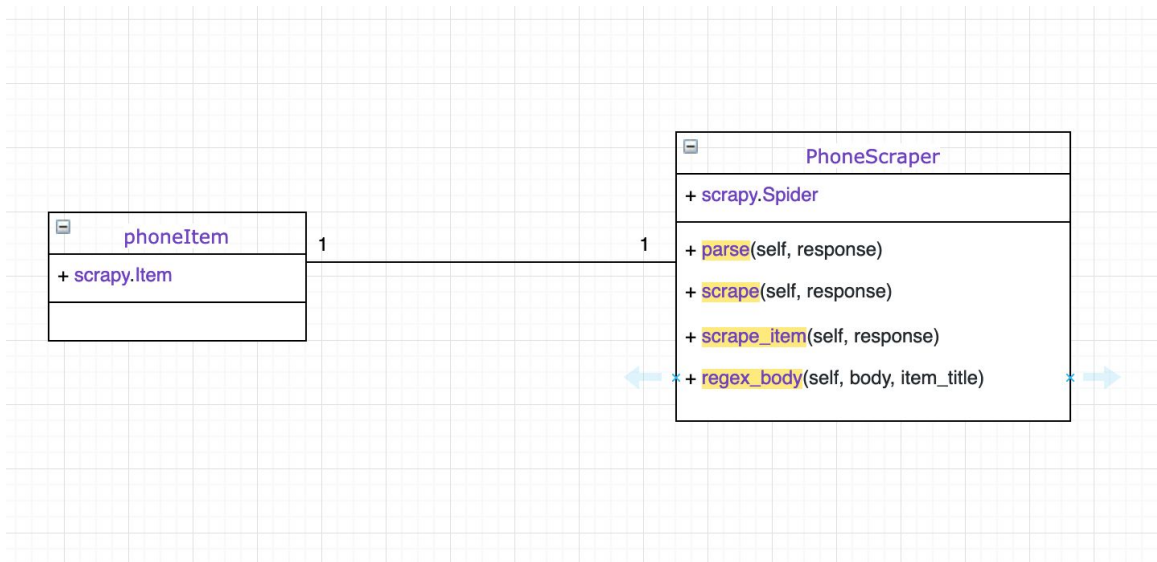


Figure 3C: Class Diagram for our web scraper

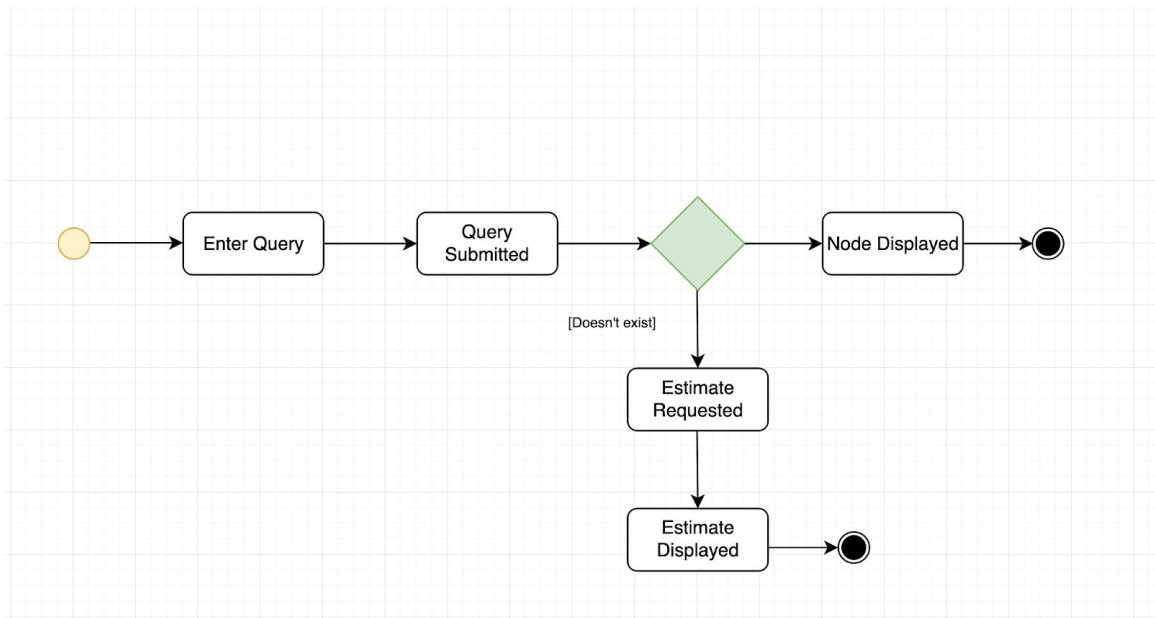


Figure 3D: Activity Diagram for Right Price

3.2 System (or Component) Functional Requirements

Table 3A. Functional Requirements

No.	User story	Functionality	Degree of Necessity
1	As a user, I shall be able to browse through items so I get to see what items are available	Functional	Essential
2	As a user, I should be able to search for items so I can find things easier and faster.	Functional	Desired
3	As a user, I should be able to submit request for price estimate for items that are not available on the application	Functional	Desired
4	As a user, I shall be able to select items to find a price estimate.	Functional	Essential
5	As a user, I shall be able to provide parameters for the selected item.	Functional	Essential

6	As a user, I shall be able to get the price estimate as a price range for the item selected.	Functional	Essential
7	As a user, I should be able to upload pictures from the camera or gallery to provide condition of the item.	Functional	Desired
8	As a user, I should be able to use the picture recognition feature so I can easily provide condition of an item.	Functional	Desired
9	As a user, I shall be able to change the parameters provided by me at any time.	Functional	Essential
10	As a user, I should be able to create an account so I save my searches for the items.	Functional	Optional
11	As a user, I should be able to log into my account so I can access my information.	Functional	Optional

3.3 Non-functional Requirements

Table 3B. Non-functional Requirements

No.	User story	Functionality	Degree of Necessity
1	As a user, I should have an overview of how to use the application on the first landing page.	Non- functional	Essential
2	As a user, I should get a new estimate every time some parameter is changed(auto- refresh).	Non-functional	Optional
3	As a user, I expect a simple user-friendly interface so I can navigate easily through the app.	Non-Functional	Desired
4	As a user, I need an application that takes under 15 seconds to give me results.	Non-Functional	Desired
5	As a user, I should have a secure authentication protocol so no one hacks my account.	Non-Functional	Desired

3.4 Context and Interface Requirements

The project is designed and implemented by using the ReactJS framework for the front-end. ReactJS is dependent on Javascript and the MVC model. The back-end of the project is developed in Python. The script for data collection as well as the machine learning algorithms are developed in Python. In order for the user to be able to run the application, a Javascript enabled browser is required. The website will be responsive, meaning it will ensure the formatting is compatible with almost all screen sizes. Thus, allowing our application to run on smartphone browsers as well. In order for the client and server to properly we will be using Python Django as our back-end service. This will allow the client to send information to the server so that the user can get the appropriate information.

3.5 Technology and Resource Requirements

Table 3C. Technologies and Resources

Requirements	Description
GitHub	A GitHub repository will be used for code management
ReactJS	Front-end framework for both web and mobile
MockFlow	Application used to create the wireframe for the prototype
MongoDB	Database will be used to store the data obtained from web scraping
AWS-EC2	Amazon AWS will be used to deploy the application
Python	Python libraries such as scikit and scrapy will be used for data collection and machine learning. We will also use testing libraries like unittest or pytest to test our software.
Python Django	Back-end development language and framework.
Documentation	Documentation should be kept up to date with instructions on using our code. Libraries like pydoc or readthedocs will be used.

Chapter 4 System Design

4.1 Architecture Design

We decided to use the MVC architecture (better known as Model View Controller) to implement our project. We used React with Redux and Webpack for the “View” part of the implementation. Our “Controllers” were implemented using Python and Django. Libraries like Scikit-learn and pickle encompassed the “Model” part of the implementation. Since we used React to implement our “View” section, we further implemented a MVVM (Model View ViewModel) architecture to create high quality front-end components that react to user response dynamically.

Our project aimed to implement MVC in a client-server setting. Our web app communicated with different server entities like the web service. The web service then communicated with the machine learning model or the web scraper as needed. Our project was hosted on AWS-EC2, which allowed us to make our application highly scalable with minimal latency.

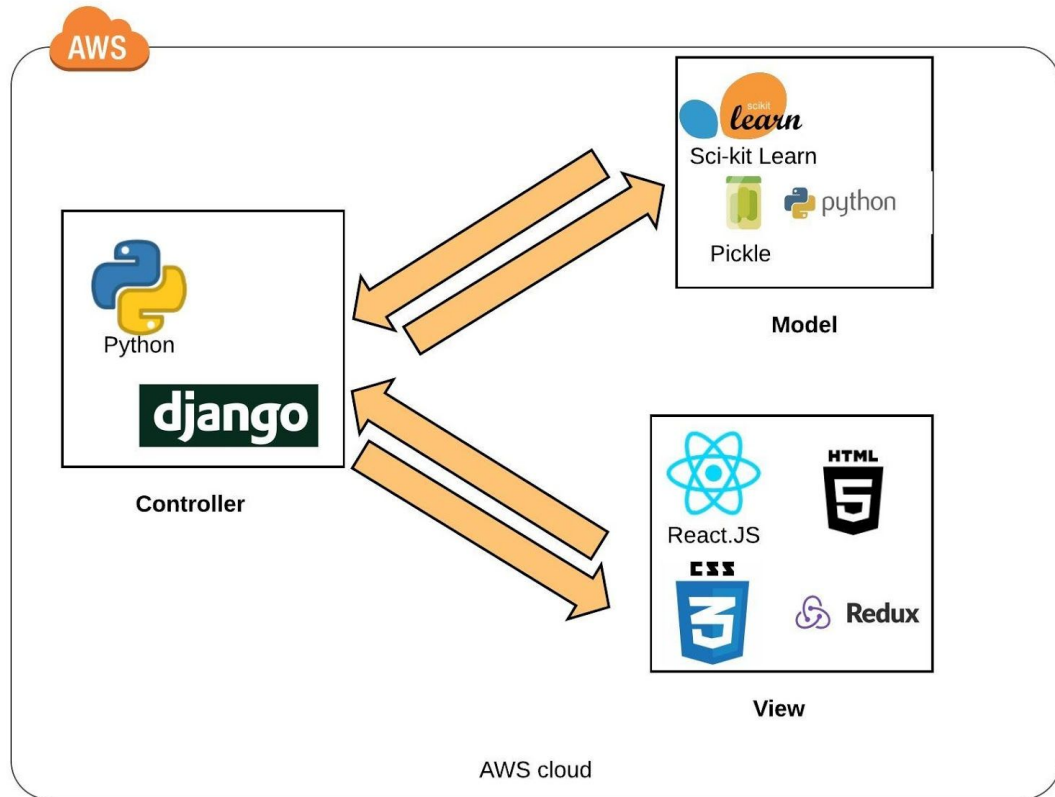


Figure 4A. Interface and Component Design

4.2 Interface and Component Design

The view was presented to users of the system. An event handler within the view tracked user behavior and sends request to the controller. Upon receiving a request, the controllers accessed or modified the model, and, depending on the user's action, the controller renders a suitable view update that displays a given version of the model. In the current implementation, React tracked user events and passed the actions to Django, the back-end service. Depending on what kind of an action Django received, Django will

directly contact the model saved with the pickle library. Once this call has been processed, Django either received the data from the model or returned an error. The results are sent back to the front-end application. If a render call is made, React updates the ReactDOM which hot-reloads components in the browser DOM (Document Object Model) without refreshing the page, all while the state is being maintained by Redux.

4.3 Structure and Logic Design

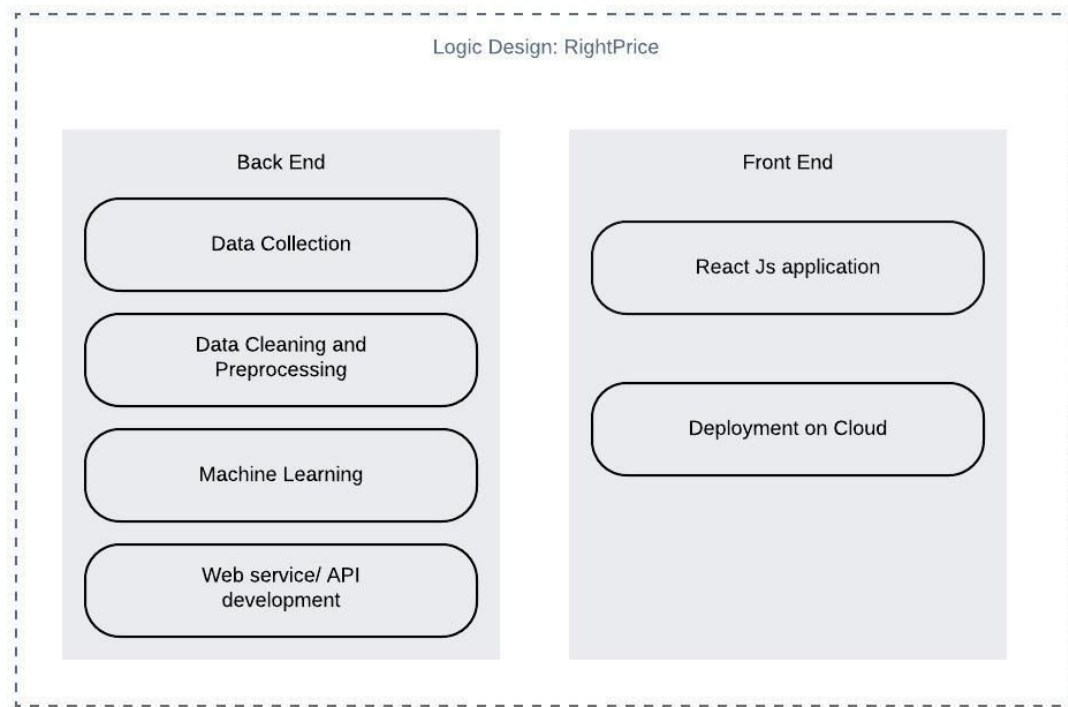


Figure 4B. Structure and Logic Design

Based on functionality, the project was divided into four major parts on the backend: Data Collection, Data Cleaning/Preprocessing, Machine Learning Model, and

the web service/API. Data flow occurred between the front-end and the back-end using an API call which is handled by a Django server. The data scraper and machine learning model run on unix cron jobs to stay updated with the most recent data. Cron jobs are scheduled once every 7 days to get a full data reset from most ad posting marketplaces. Based on the new ad postings that are collected through the web scraper, the machine learning model is trained again. Through the automated process, the trained model is then deployed to the web service, which can then be accessed by a user.

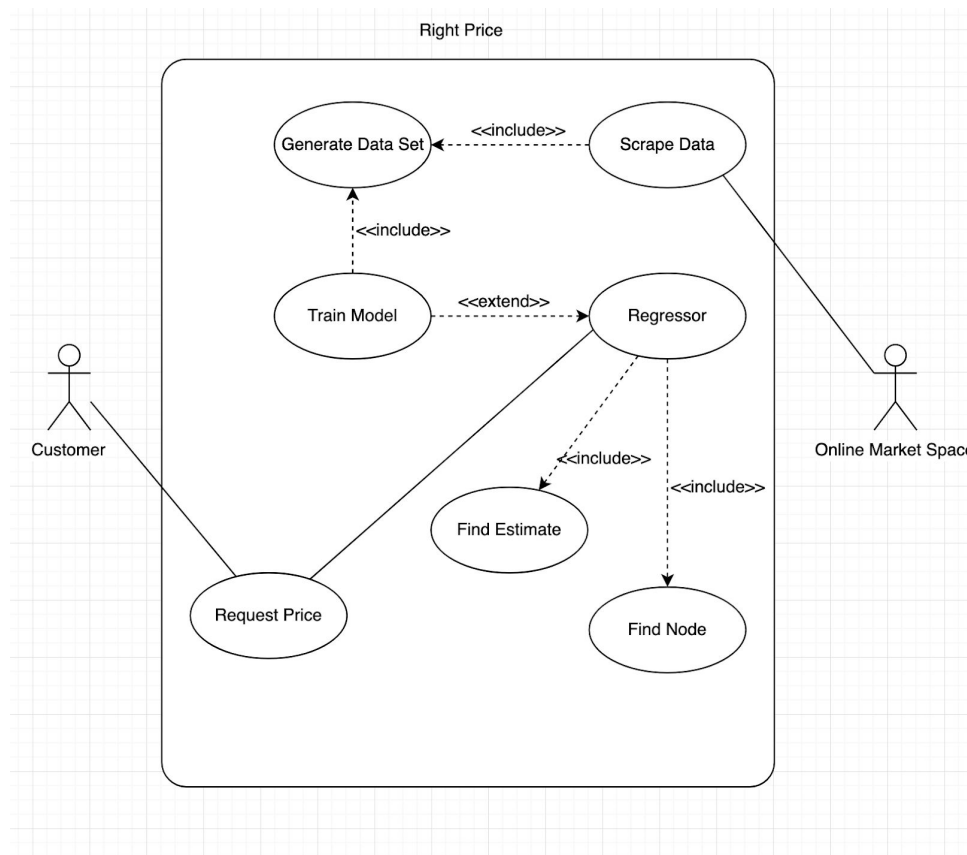


Figure 4C. User Access and Application Flow

4.4 Design Constraints, Problems, Trade-offs, and Solutions

In this section, we examine the design constraints considered when designing our software, and the trade-offs and solutions made to solve or work around those constraints.

4.4.1 Design Constraints and Challenges

Some of the design constraints involved in the design of our software included dealing with how advertisements were posted on online marketplaces, the vernacular used in different areas in the United States, the current culture of shopping online for used goods, etc.

For example, advertisements were posted in different ways depending on the online marketplace. In the case of Craigslist, users posted their device for sale and tagged various metadata about the device, such as condition, color, carrier, operating system, or manufacturer, or the user may leave these tags out. This was different than on Ebay where the specification of the device are required for the posting. This led to our design focusing on pulling data from one source, cleaning the data as much as possible, and feeding the cleaned data into our machine learning model.

Another example, as we pulled data from different areas in the United States, we had to deal with differences in language. As an example, several posts in the Bay Area tended to be less wordy and have far less detail with respect to the specification of the device. In contrast, posts in New York used more sophisticated language with more

detail. Our design had to be able to pull data from both areas of the United States accurately.

Another main design constraint on the technical side of things dealt with the machine learning models. We tried to design a model that was efficient and produced accurate and precise results as much as possible. This was incredibly tough as used device prices varied heavily throughout different regions and with dirty data.

One of the biggest challenges for us was to apply machine learning to solve our problem. Most of the data was in non numeric format like brands (for example, Apple or Samsung) which have no relation between themselves. Because machine learning works by making correlation between the data, we had to come up with a strategy to preprocess the data, so that our machine learning model could be used to fit our dataset and then later be queried for predictions.

Another problem was to create an efficient machine learning model which can provide accurate prediction with low latency. Applying machine learning model on large datasets takes significant time. If a model has to fit every time a user asks for prediction, it would take about one minute of time to return a prediction. We had to design a solution to reduce this latency and deploy an already fit model.

Finally, consideration was given to how the user interface of our application will present data to the user. A good user interface should present the result of the data

computation succinctly and neatly. Additionally, a good user interface should be easy to use and intuitive.

4.4.2 Design Solutions and Trade-offs

To solve or work around some of these design constraints, our team designed changes to our software or developed new methods to clean data. For example, when facing the challenge of how advertisements were posted on Craigslist, we designed a way to pull as much data as possible from every post under the smartphone postings section. Then, we cleaned the data manually as much as possible before feeding the data into our machine learning model. This helped immensely in generating a model that could predict prices accurately and with great precision. This conveniently also helped with our secondary design goal of designing a machine learning model with accurate and precise results. The process of “cleaning data” was also necessary to decode how posts were created in various regions of the United States.

To solve the problem of non numeric data, we used OneHotEncoder, a tool from Scikit-learn library for data preprocessing. OneHotEncoder converts non numeric categorical data into binary/numeric data. Then, machine learning models were applied to the non numeric categorical data. Here is an example from data where OneHotEncoder was used to convert categorical data into numerical data.

Table 4A. Regular categorical non numeric data in the original dataset

Brand	Price
Apple	600
Samsung	525
Google	295
LG	200
Apple	345

Table 4B. Numeric data preprocessed using OneHotEncoder

Apple	Samsung	Google	LG	Price
1	0	0	0	600
0	1	0	0	525
0	0	1	0	295
0	0	0	1	200
1	0	0	0	345

K-Nearest Neighbors algorithm was chosen out of various other algorithms available because it could work well with our problem although we did consider using other algorithm solutions like multiple linear regression models. It was a regression problem which needed some classification in the back and KNN regressor algorithm was best suited for such problems. For reducing the latency and avoiding to refit the model every time for a prediction, we saved an already fit model using pickle library in Python. Then, we deployed the saved model with AWS.

The user interface design constraint of our software was solved by working closely with all team members to design wireframes before implementing the final design. This

allowed the team to provide continuous feedback as an ideal user interface was designed. Finally, when a user interface design was approved by all members of the team, it was implemented with Javascript and ReactJS tools as described later in this report.

Chapter 5 System Implementation

5.1 Implementation Overview

Using Python and libraries like Scrapy, we designed an algorithm that scraped data from a public marketplace like Craigslist and stored that data into a CSV file. After running the data collection algorithm on multiple locations in the U.S., we gathered about 13,000 data points. Then, we used some Python scripts to clean and preprocess the data. Next, we fed the cleaned and preprocessed data through a K-Nearest Neighbors (KNN) regressor algorithm. The trained model was then stored using tools within the pickle library. Next, we stored the model in an application wrapper using the Django framework to make up our back-end application. Finally, we were able to predict the price of a smartphone using an API call from the front-end application containing several parameters such as the condition, brand, model, etc. of the smartphone.

In order to implement the front end, we used ReactJS. The created framework was used to handle all of the logic behind what the user sees and to handle user interaction. This made our API accessible to the average user.

5.2 Implementation of Developed Solutions

In the data collection phase of our project, we used the Spider tool from the Scrapy library in Python. We ran the algorithm on multiple metropolitan locations within the United States to get a bigger dataset. Figure 12 shows a snippet of code that controlled

which locations in the United States were scraped. All data was collected from the corresponding Craigslist website for that location.

```
[ 'https://sfbay.craigslist.org/d/cell-phones/search/moa',  
  'https://lasvegas.craigslist.org/search/moa',  
  'https://losangeles.craigslist.org/search/moa',  
  'https://sandiego.craigslist.org/search/moa',  
  'https://orangecounty.craigslist.org/search/moa',  
  'https://palmsprings.craigslist.org/search/moa',  
  'https://sacramento.craigslist.org/search/moa',  
  'https://newyork.craigslist.org/search/moa',  
  'https://seattle.craigslist.org/search/moa',  
  'https://chicago.craigslist.org/search/moa',  
  'https://miami.craigslist.org/search/moa',  
  'https://washingtondc.craigslist.org/search/moa']
```

Figure 5A. Code snippet showing locations where data was scraped

Based on manual screening of outliers in the dataset, most of the ad postings under \$11 or more than \$1100 were ignored as they were spam most of the time. Our spider algorithm first looked for attribute tags related to the ad posting such as model, price etc. While the algorithm received some data from the tags, much of the required data was missing. Then, the spider algorithm scanned the title for more information using regular expressions (reg-ex). If some of the required features were still missing, the spider searched the ad's URL and scanned the actual body of the ad posting to grab more information. Repeating these steps for all smartphone listings, we were able to collect a little over 12,000 data points. Figure 13 shows the result of running a script with our spider algorithm.

```
21:30:45 [scrapy.core.engine] INFO: Closing spider (finished)  
21:30:45 [scrapy.extensions.feedexport] INFO: Stored csv feed (12291 items)
```

Figure 5B. Statistics of collected data points

We then cleaned the data by automatically removing the ad postings with spam keywords like “car”, “mount”, “watch”, “headphones” etc. to get a better dataset. Some of the postings were missing certain attributes but had others attribute that could be used to fill in missing data. Our algorithm used these inferences to fill in missing attributes. For example, if the ad posting was missing the brand attribute, but it had “iPhone” in its title and model field, our algorithm filled in “Apple” for its brand attribute. Figure 14 shows a code snippet of this logic in our algorithm.

```
if item['brand'] == None and item['model'] != None:
    model = str(item['model']).lower()
    if re.search("(iphone)", model):
        item['brand'] = "apple"
```

Figure 5C. Code Snippet of Data Cleaning

Table 5A. Table showing a few rows of collected and cleaned data

BRAND	COLOUR	CONDITION	CONTRACT	MEMORY	MOBOS	MODEL	PRICE
Samsung	gray	like new	Sprint	256	android	S8 PLUS	\$270.00
Blu	red	like new	Unlocked	16	android	Y85	\$50.00
Samsung	Black	Good	Verizon	32	android	S10 plus	\$745.00
Apple	gray	like new	T-Mobile	64	apple iOS	iPhone 8 Plus	\$400.00
Samsung	white	excellent	Unlocked	64	android	NOTE 8	\$70.00
Apple	white	like new	Unlocked	256	apple iOS	XR	\$400.00
Apple	white	like new	Unlocked	64	apple iOS	6 PLUS	\$220.00
Apple	black	good	Unlocked	32	apple iOS	6S	\$150.00
Apple	rose	good	Unlocked	32	apple iOS	6	\$110.00
Apple	black	excellent	Unlocked	32	apple iOS	6 PLUS	\$155.00
Apple	rose	excellent	At&t	64	apple iOS	6	\$140.00
Apple	black	good	Verizon	64	apple iOS	6	\$135.00
Apple	silver	excellent	At&t	128	apple iOS	IPHONE 7	\$225.00
Apple	white	Good	Unlocked	256	apple iOS	XS	\$600.00
Apple	white	like new	Unlocked	128	apple iOS	IPHONE 7 PLUS	\$275.00
Apple	white	like new	Unlocked	64	apple iOS	iphone 8	\$365.00
Samsung	gray	excellent	T-Mobile	128	android	S10 PLUS	\$180.00
Apple	white	excellent	At&t	128	apple iOS	IPHONE 7 PLUS	\$400.00
Samsung	black	excellent	Unlocked	64	android	S10	\$199.00

We also performed some data preprocessing on the cleaned data to convert non-numeric categorical data into binary numeric data using OneHotEncoder from the Scikit-learn library.

```
from sklearn.preprocessing import OneHotEncoder
# all strings parse to binary arrays that mean presence or not
enc = OneHotEncoder(handle_unknown='ignore', sparse=False)

r = np.reshape(dataset[:,0],(10484,1))

brand=enc.fit_transform(r)
color=enc.fit_transform(np.reshape(dataset[:,1],(10484,1)))
condition=enc.fit_transform(np.reshape(dataset[:,2],(10484,1)))
contract=enc.fit_transform(np.reshape(dataset[:,3],(10484,1)))
mobOs=enc.fit_transform(np.reshape(dataset[:,5],(10484,1)))
model=enc.fit_transform(np.reshape(dataset[:,6],(10484,1)))
```

Figure 5D. Code snippet showing data preprocessing using OneHotEncoder

We used the K-Nearest Neighbors as our choice of machine learning algorithm to process the data gained by scraping online marketplaces. This algorithm is typically used for classification on regression.

KNN identifies k-nearest neighbors of an estimate point in a pool of pre-existing training vectors. For example, after training the KNN model with k as 3, when we test it by predicting the price of a smartphone, it returns 3 nearest neighbors for that smartphone measuring the Euclidean distance between those points in a multidimensional space. Using the neighbor's prices, we can predict the price and the price range for the target smartphone.

We used Python's pickle library to save a trained model into a SAV (.sav) file which could later be loaded easily and queried at any time with no latency. We used a Django framework in Python to make a backend web service/API to work as server for our application. The service was hosted on an AWS EC2 instance which can be used at ec2-3-15-200-193.us-east-2.compute.amazonaws.com.

An API call can be made using a browser using the following example:
<http://ec2-3-15-200-193.us-east-2.compute.amazonaws.com/predict2?brand=apple&color=black&condition=excellent&contract=unlocked&memory=256&mobos=apple&model=iphone x>

For the above given request, the following result is returned.

```
{  
  "predict": "402.0",  
  "nn": "[500. 570. 250. 575. 115.]"  
}
```

Figure 5E. Figure showing results received when the API is called

For the queried smartphone (i.e. Unlocked Apple iPhone X in excellent condition in black/space gray with 256 GB of storage), it returns a predicted price as well as prices for the 5 nearest neighbors.

For the front-end, implementation was fairly straightforward. We wanted to ensure the layout of the page was intuitive and usable for the user, thus we ended up going for a single page application. The benefit of ReactJS is that it is able to quickly re-render components onto the Document Object Model (DOM), which is essentially what the users sees. The next step was ensuring that the website was responsive and usable, even on smaller screens. The final step was ensuring that errors were handled properly. In cases where the user did not fill out the entire form, it was important to inform the user so they could fix it before continuing to use the application. After all the errors were properly handled and the form properly filled out, the backend API is called with the respective parameters and a predicted price is shown to the user. The below figures show how the form may be filled out and the result of the API call from the front-end application.

RightPrice

GETTING STARTEDSEARCHRESULTSABOUT THE TEAM

Search for an item:

Phone type:

Color:

Condition:

Contract:

Memory:

Mobos:

Model:

Apple

Space Grey

Good

Unlocked

64GB

Apple iOS

iPhone 8 Plus

SEARCH

Figure 5F. User Entry Form

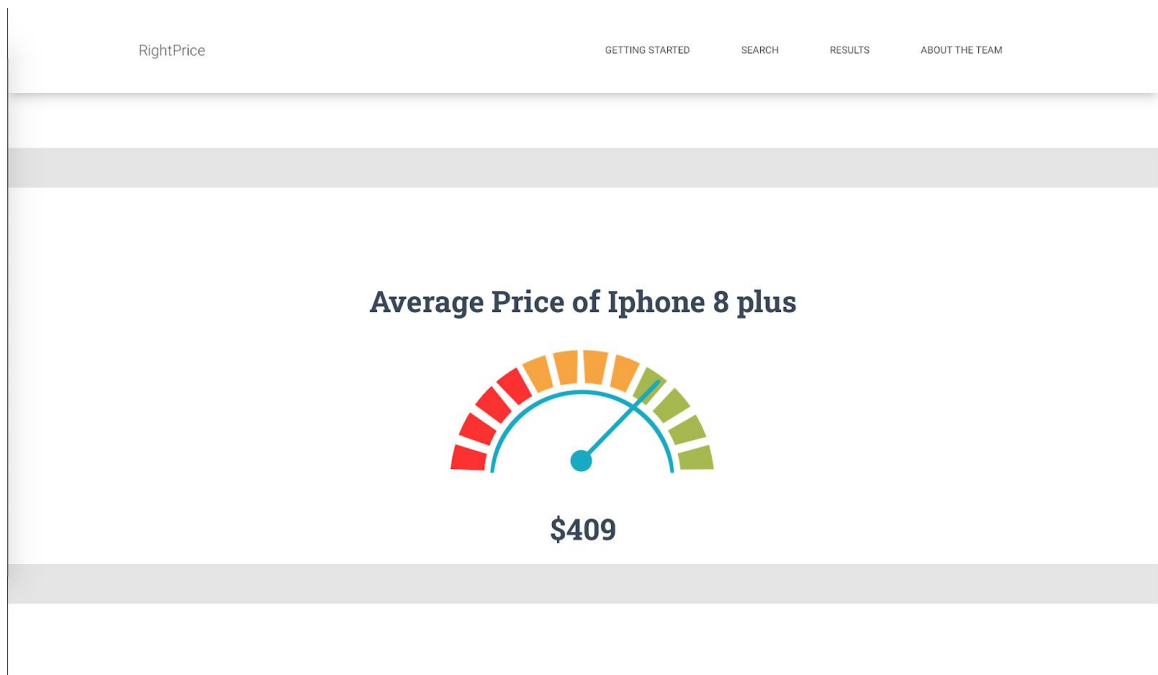


Figure 5G. User Results Page

5.3 Implementation Problems, Challenges, and Lesson Learned

Implementation of our machine learning model was hindered by the lack of a robust feature set that could have made the process much easier. Our dataset consisted of non-metric features such as color, brand and model name. Such features made it harder for us to write an algorithm that would either classify or perform regression. For our current iteration we scraped values for first party description pages and utilized some of the numeric data (for instance the screen size of a phone, pixel density, battery capacity, etc.)

It was difficult to find an efficient solution for the machine learning model. We considered using other simple algorithms such as multiple linear regressions for the prediction models. However, it didn't work as results from various models could not be combined. Moreover, it would have been hard to make a real correlation between predictions provided by various linear regression models for different features.

In terms of the front-end development, we used a custom library for certain UI elements so we would not have to create them from scratch. However, some of these libraries have poor documentation making the implementation much more difficult than it should have been. Finally, integration of the back end and front end was slightly challenging since no one in the group had much experience in this topic. Online resources and tutorials helped ensure that the process went by smoothly.

Chapter 6 Tools and Standards

6.1. Tools Used

The following tools were used to help facilitate team meetings, track our progress, and write software:

Google Drive:

We used Google Drive because of its ubiquity among SJSU students especially among our team. Its collaboration tools enabled us to work efficiently on documents like this project report, diagrams, and other data. Google Drive also housed all of our data necessary for course assignments so that it was accessible to everyone, even while away from a computer.

WebEx:

This meeting software allowed us to hold team meetings remotely. This proved especially useful as there were several times when not all of our team could be on campus. This tool also helped us conduct meetings with our advisor remotely.

iMessage:

Most communication occurred through Apple's messaging platform. Since all of us own Apple products, iMessage was ubiquitous among the team and aided in quick

communication especially when urgency was required. iMessage also made it simple to share links, pictures, and documents throughout the project timeline.

GitHub:

This version control software proved essential to code collaboration. Because we had multiple team members working on different parts of the project at different times, GitHub and git tools in general helped to ensure progress was saved and code was merged correctly and safely. GitHub also provided a way for our advisor to keep tabs on our project progress. GitHub also offers tools for project planning that proved beneficial during the project planning phase. We chose to use GitHub over other Git platforms as all teammates were familiar with the platform already and had user accounts with GitHub.

VSCode:

This lightweight but powerful text editor offers a marketplace that supports all kinds of file types. This combined with powerful software writing tools made it the obvious choice for all teammates to use to write code. Additionally, VSCode offers git tools to easily pull and push code to GitHub. We used VSCode to write almost all of our code and our testing suite.

Canvas:

Canvas is a tool offered by SJSU to help manage learning at the university. It allowed the team to be grouped together and be recognized by the course instructor and our advisor. It also provided a way for us to submit assignments to be graded by the respective grader assigned to each assignment. We chose to use Canvas as a result of its ubiquity throughout SJSU. Furthermore, using Canvas was a requirement to pass this course.

The following tools were used during the data collection phase of our project:

Python Scripting:

Due to the repetitive nature of collecting data from online marketplaces like Craigslist, the team found it useful to write a Python script to visit each user post and collect relevant data through web scraping. The Python scripts we wrote were powerful and fast enough to collect up to 10,000 data points in a few minutes. Furthermore, we were able to reject data through simple validation logic. The scripts were also adaptable and flexible enough such that we could change what location to collect data from, which was especially useful for Craigslist ads in different parts of the United States.

Scrapy:

Scrapy is an open-source and collaborative framework for scraping data from websites. Scrapy helped us write our own web spider that automatically combed through

a collection of web pages based on a specific parameter (for example smartphones). This combined with our Python scripts enabled us to write minimal code to begin web scraping. We chose to use Scrapy due to its expansive community, which provided help and support when learning how to use the tool, and its ease of use. With Scrapy, we easily collect the data we needed. Scrapy was also easy to install on our personal machines and had a ton of documentation on how to use the framework. All data was stored in a CSV format as Scrapy makes it easy to store data this way.

The following tools were used in the machine learning portion of our project:

pickle:

Pickle is a library that can save a machine learning model into a file. It does this by serializing a Python object structure (our trained ML model) and de-serializing the file into a Python object when needed. This way, we can query the file instead of re-training the machine learning model every time we wanted to predict a smartphone's price. We chose to use pickle over something like JSON or the marshal library because it integrated well into Python, worked well across different Python versions, and worked with user-defined classes.

Scikit-learn:

This massive machine learning library offers tools for data mining and data analysis. As our project required analyzing data collected off of online marketplaces, scikit-learn proved to be a great asset. Scikit-learn also has a fantastic community with plenty of resources as we learned how to create a machine learning model. Scikit offered specific tools that we used for our machine learning model:

KNN model:

This model finds the nearest neighbors by plotting data points into a virtual space and determining the Euclidean distance and returns a prediction when the model is queried. The prediction is based on an average of the nearest neighbors. Scikit-learn offered easy APIs to build our own KNN model.

OneHotEncoder:

We used this to convert our non-numeric data into binary data arrays. This allowed us to put our data into something Scikit-learn's machine learning models could understand.

Django:

Django is a Python framework for backend web applications. We chose Django as Django works extremely well with Python. Any other frameworks may have

unnecessarily complicated our application (for example, NodeJS meant integrating a second language.) We used this to build our API and create our backend application to run as a web server for our price estimator.

AWS-EC2:

AWS-EC2 provided by Amazon served as an actual hardware server to host our web application. EC2 stands for elastic compute cloud. EC2 provided us with a virtual space to run an instance of our web application so that anyone could access it. This was extremely helpful as the alternative meant the team would need to create our own hardware to host our application. AWS was chosen as it was the most reliable, scalable, and ubiquitous web service.

The following tools were used in the front-end development for our project:

ReactJS:

ReactJS is a Javascript library used for creating interactive websites. We chose this because our team was mostly familiar with using ReactJS from previous coursework. ReactJS offered an easy way to write components and stable code. Because of its accessibility, ReactJS was chosen to build the front-end of our application. It helped used build a system to take user input on a website, validate these inputs, and return a result.

Creative-Tim:

In an effort to save time, yet still build a beautiful web application, the team chose to use a template from Creative-Tim. Creative-Tim offers several tools and templates that helped us develop a first draft for our user-facing web application. It also offered simple tutorials and documentations to help us get started.

6.2. Standards

Some examples of engineering standards using throughout this project are as follows:

PEP 8:

In order to have consistently formatted code that was easy to read by all team members, we chose to follow the PEP 8 style guide for Python. PEP 8 is a coding standard that dictates how code should be formatted. This includes but is not limited to naming conventions for variables, functions, and files, tabs vs. spaces, the maximum length of a line of code, the amount of blank lines after function headers or a module, and comments. This was especially beneficial as code remained neat through integration with VSCode.

Agile:

We did our best to follow some form of software development methodology. For our purposes, we selected a blend of Agile and traditional software development

processes. This enabled the team to have weekly deliverables that showed our progress throughout the project. This had the benefit of exposing any issues in our design up-front instead of at the end of the implementation in the testing phase. We also used Agile communication practices like weekly meetings to update each other on goals and progress. We tried to break down larger project goals into smaller ones that could easily be completed on a weekly basis. Agile methodology also proved helpful when someone faced a blocker as we could turn to the team lead or another teammate for help.

IEEE 12207 - Systems and Software Engineering - Software Life Cycle Processes:

This standard establishes a framework for software life-cycle processes. This standard establishes certain processes for the development, operation, maintenance, and disposal of software products. While our team mainly focused on the development and testing, it was useful to review some of the guidelines listed in this standard as outlined by IEEE. Ultimately, the standard outlines how outside stakeholders are involved in the development of software with the ultimate goal of achieving customer satisfaction. In our case, we worked to satisfy the CMPE department with our software quality as well as the course director with our advisor acting as a stakeholder in our project.

JSON:

JSON or JavaScript Object Notation was used very heavily in the implementation of our front-end application. It was also used to send data between the back-end and front-end application. JSON is a standard formatting for lightweight data-interchanges. It defines a way to format data for use in Javascript applications. However, it can be used in other languages or application because it follows a standard formatting. Therefore, our back-end API serves data requests in JSON format with all necessary data. Our front-end application parses this information and presents the result to the user. Furthermore, when a user selects which smartphone they're looking for on our website, each drop-down list is populated with data from a hardcoded JSON file containing all smartphone data and specifications.

Chapter 7 Testing and Experiment

7.1 Testing and Experiment Scope

The test process used to test RightPrice included testing on multiple levels such as unit testing, integration testing, system testing and acceptance testing. Since the development of RightPrice was done in three major segments, we had to perform regular unit tests on all the units. We also used both white box and black box testing to test the application.

White box testing is a method of testing in which the tester has knowledge about the internal structure and the code of the program. On the other hand black box testing is a method in which the tester doesn't have any knowledge about the internal structure and code. Black box testing was used to ensure that our website functioned properly, and was easy to navigate for the user. White box testing was used to ensure that the logic of the software worked as intended and each element's functionality is what was intended. This was only possible with white box testing, as we knew what the intended behavior should be.

Unit tests mainly focused on finding the bugs among all the units. In the case of web scraping, testing encompassed testing the quality of the scraped data points and to ensure the algorithm was cleansing the data properly. Unit testing for the machine learning part of the RightPrice was done by running various tests for scoring our machine

learning model. The front end's unit testing mainly comprised of removing any GUI bugs such as overlapping assets, page responsiveness, rendering and dismounting components.

Integration testing focused on getting the above-mentioned units working together properly without causing any bottlenecks. Backend development and testing was done during this process as it played an important role in merging the different elements of RightPrice. Test needed to be done with the web scraper and the machine learning algorithm was minimal as they only shared a CSV. Access clashes were the only things that were to be checked as a part of integrating these units. Integration of the frontend to the newly integrated module mainly focused around the backend development as the frontend behaved in accordance to the responses from the backend. Backend integration with the logical units of RightPrice mainly encompassed testing the accessor and mutator functions of the backend to the machine learning algorithm.

Test focus for System Testing included testing for any possible bugs exhaustively in a given time frame to make sure product quality was up to par and RightPrice was ready for launch. Unit tests and Integration tests were all run another time after the final system integration.

Acceptance testing has been completed, and it focused on looking at RightPrice and evaluating whether it complied with the user requirements. In order to achieve this, we released our product to our group members and friends. We wanted to ensure that the application was user friendly and helpful. After allowing a couple of people to test the

application, our results were mainly positive. The application was very straight forward and everyone had no problem navigating it. In terms of usefulness, most users agreed that it helped give them an estimate of a price for the smartphones they want to buy or sell. Thus resulting in making their online shopping much easier, as they know what an average price should be. Overall, acceptance testing was very successful and all the users found the application useful.

7.2 Testing and Experiment Approach

Table 7A. Scope of the Tests and Expected Results before testing

No.	Scope of the Test	Expected Result
1	As a user, I shall be able to browse through items so I get to see what items are available	User is able to browse possible phones
2	As a user, I shall be able to select items to find a price estimate.	User is able to select a particular phone model
3	As a user, I shall be able to provide parameters for the selected item.	User is able to selected parameters such as: Memory, Color, etc.
4	As a user, I shall be able to get the price estimate as a price range for the item selected.	User is able to get a value after selected parameters
5	As a user, I shall be able to change the parameters provided by me at any time.	User is able to change parameters at any time

Unit tests were performed on various parts of the project i.e. web scraper, data cleaning part, machine learning algorithm, backend API and the front-end project. The following describes the scope of the tests performed on the units of the application.

Our web-scraper was tested mainly using black-box testing. A link to a website was presented to the algorithm and results from the algorithm were inspected to ensure the algorithm was dropping any unnecessary content and sanitizing all the useful content based on a reference.

For testing our machine learning model, we used the coefficient of determination R^2 (r squared) scoring method. This method provides how far were the predictions to the actual values/ plotted line. The scoring method compares the values predicted with the plotted regression line. Root-mean-square Error method is one of the most used methods to score machine learning models. It explains the relationship between the machine learning model and the dependent variable, on an easy-to-interpret scale of 0 to 1 or 0 to 100 percent. According to Scikit-learn documentation, “The coefficient R^2 is defined as $(1 - u/v)$, where u is the residual sum of squares $((y_{\text{true}} - y_{\text{pred}}) ** 2).sum()$ and v is the total sum of squares $((y_{\text{true}} - y_{\text{true.mean()}}) ** 2).sum()$ ” (Pedegrosa, 2011, p. 2825) [6].

Front end was mainly tested through peer programming. A team of two worked on development and testing. The programmer would implement the frontend and do some testing on his part and a dedicated tester would look for any bugs and pass back to the

programmer to fix the bugs. The testing and fixing process was done until the Front End did not have as many visible bugs.

7.3 Testing and Experiment Results and Analysis

After running the tests using various techniques defined above, here are the results that we got.

Table 7B. Actual results after testing

No.	Scope of the Test	Expected Result	Actual Result
1	As a user, I shall be able to browse through items so I get to see what items are available	User is able to browse possible phones	User is able to browse through potential phone models
2	As a user, I shall be able to select items to find a price estimate.	User is able to select a particular phone model	User is able to select a particular phone model
3	As a user, I shall be able to provide parameters for the selected item.	User is able to selected parameters such as: Memory, Color, etc.	User is able to choose parameters to refine their search
4	As a user, I shall be able to get the price estimate as a price range for the item selected.	User is able to get a value after selected parameters	User gets a value returned after selecting parameters
5	As a user, I shall be able to change the parameters provided by me at any time.	User is able to change parameters at any time	User is able to update parameters at any time

The web scraper was able to go through 2100 ads in an average time on 18 second. The yield of a scrape was about 1800 ads. There was no data point repetition and all data points that were empty were filled up in the results.

The data was split into 80% for training purposes and 20% for test purposes. The scores from testing our machine learning model using R2 (r squared) scoring method came out to be about 87% for training data and about 80.5% for test data.

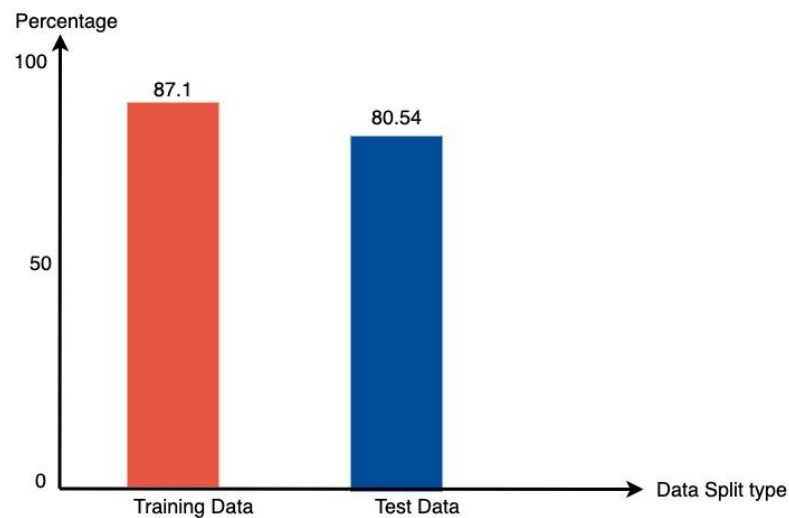


Figure 7A: Test Results for Training Data and Testing Data

Chapter 8 Conclusion and Future Work

When we set out to build RightPrice, we endeavoured to build a usable project that could solve a real problem we saw in the world. That is, we sought out to build a solution to the complexity in navigating the used marketplace. We saw how difficult it was to buy and sell a used good without accurate price information about its market price. Furthermore, bad actors could use the uncertainty of the marketplace to scam people by asking for a higher price than a used item was worth or offering a low price for something someone was selling. Our idea was to build a project that could collect existing price data for used goods on online marketplaces, teach a machine learning model how to predict prices based on a variety of factors, and return the average market price to a user. Ultimately, we focused on smartphones to narrow the scope of the project. We focused on a few characteristics like age, condition, brand, model, color, among others to determine the price of a requested smartphone. Our team successfully delivered a working project and a report detailing our efforts. In the process of researching the various machine learning solutions to solve this problem, we found out the K-Nearest Neighbors(KNN), is best solution when regression problem is to be solved by categorical data.

As for the future work, there are numerous ways our project could definitely be improved and expanded upon. In terms of improvements, we could improve the machine learning model and the data collections scripts. Ideally, with proper resources and

funding we should be running our data collection scripts every two weeks or so. This ensures that our data set remains up to date with newer models and prices. As the data set is updated routinely, the machine learning algorithm would also have to be retrained. This ensures that the model uses the newest data sets and is able to accurately predict the prices of newer models as time progresses. Another update to the data collection would be to improve detection of fraudulent advertisements. These ads can lead to an incorrectly trained machine learning model, which could be exploited by attackers. The machine learning algorithm by itself can be improved upon by using a more comprehensive feature list.

GitHub repository with project code: <https://github.com/manmeet-gill/RightPrice>

Front-end application: <https://master.dm8gjsj35mxwk.amplifyapp.com>

References

- [1] Asim, Muhammad & Khan, Zafar. (2018). Mobile Price Class prediction using Machine Learning Techniques. *International Journal of Computer Applications*. 179. 6-11. 10.5120/ijca2018916555.
- (Summary: The work in this article outlines how various technical specifications for a phone from www.GSMArena.com can inform a price classification for that phone using machine learning techniques.)*
- [2] Deloitte. (2017). Used Smartphones: \$17 Billion market you may have never heard of. *Deloitte Network Global*, (4), 3. Retrieved from <https://www2.deloitte.com/content/dam/Deloitte/global/Documents/Technology-Media-Telecommunications/gx-tmt-prediction-used-smartphones.pdf>.
- (Summary: This article explains the large market of used smartphones. As smartphone prices continue to rise and consumers wish to upgrade their phone, old smartphones are offloaded to other users in the market for a cheaper phone.)*
- [3] Fabian, Pedregosa. (2011). Scikit-learn: Machine learning in Python. Cambridge, Mass.]
- (Summary: This article explains how tools in scikit-learn can be used in various applications that require machine learning. The article goes over simple examples and the potential results you can expect from the tools.*

[4] Glez-Peña, D., Lourenço, A., López-Fernández, H., Reboiro-Jato, M., &

Fdez-Riverola, F. (2014). Web scraping technologies in an API world. London] :

doi:10.1093/bib/bbt026

(Summary: This article explains web scraping strategies that can be used to collect data from websites, especially using APIs developed by large companies like Google, etc.)

[5] Harrison, Onel. “Machine Learning Basics with the K-Nearest Neighbors

Algorithm.” Medium, Towards Data Science, 14 July 2019,

www.towardsdatascience.com/machine-learning-basics-with-the-K-Nearest-neighbors-algorithm-6a6e71d01761.

(Summary: This article explains the basics of machine learning using a common model and algorithm: K-Nearest Neighbors. The article offers an introductory tutorial to users looking to get started with machine learning in their applications.)

[6] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830,

2011. <http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html>

(Summary: This article written by the authors of Scikit-learn details how to use its various tools. It explains the ease of use, performance, documentation, and other goals when creating the library.)

[7] Poursaeed, O., Matera, T. & Belongie, S. Machine Vision and Applications (2018)

29: 667. <https://doi.org/10.1007/s00138-018-0922-2>

(Summary: This article explains the problem of estimating the market value of real estate. Companies like Zillow have begun to use machine learning to build a prediction model for real estate using proprietary formula.)

[8] Wu, X., Zhu, X., Gong-Qing Wu, & Ding, W. (2014). Data mining with big data.

New York, NY : doi:10.1109/TKDE.2013.109

(Summary: This article goes into depth about how to use data mining techniques and the solutions one can hope to achieve.)

Appendix: Data on Smartphone Prices

This appendix contains a few visualizations showing collected data trends in smartphone pricing in the used market. This appendix is supplementary to the report and not a main deliverable for our project.

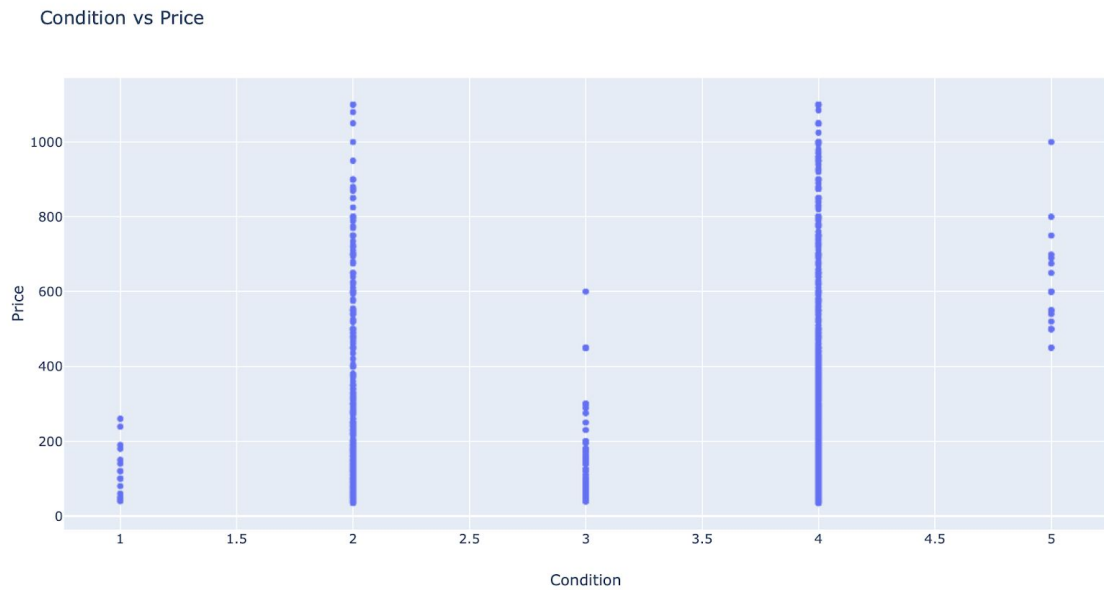


Figure AA: Price vs. Quality on a scale from 1 to 5

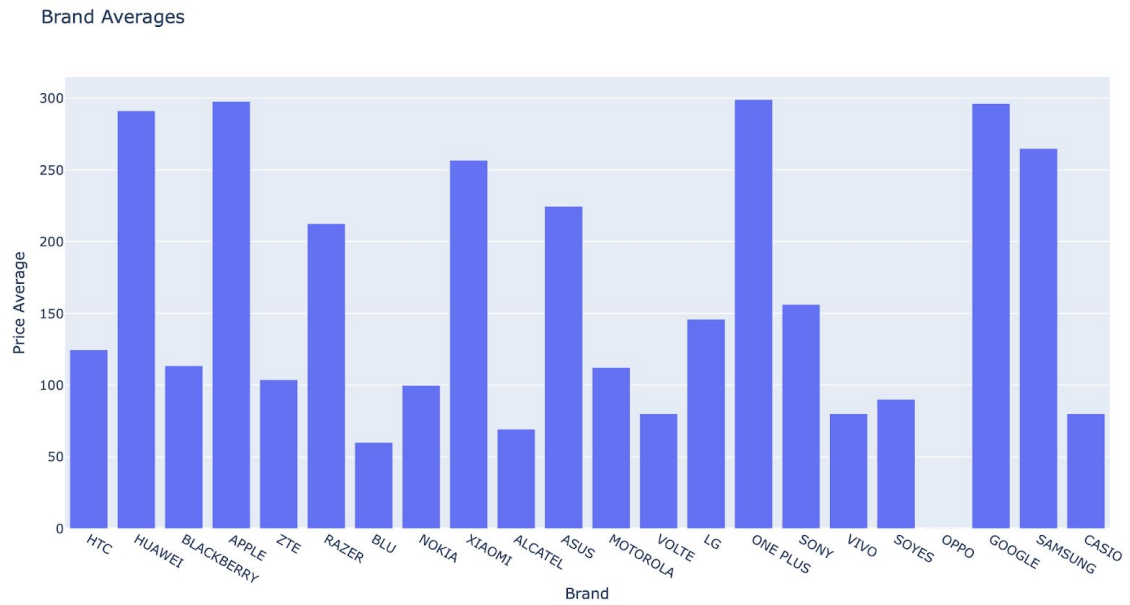


Figure AB: Average price of devices based on manufacturer