

**A**

**PROJECT REPORT**

**ON**

“​**​** ChatBot To Bargain Price For Woo-commerce Portal”

For the subject Lab **II Project Phase II**

Submitted in partial fulfillment of the requirement for the award of

**Bachelor of Engineering**

**In**

**Computer Science and Engineering**

**Solapur University, Solapur**

By

|  |  |  |
| --- | --- | --- |
| Name | Roll. No. | Exam Seat No. |
| Chayan Bobra | 09 | 807028 |
| Kartik Jain | 10 | 807129 |
| Kunal Burgul | 11 | 807046 |
| Sujay Khandare | 12 | 807110 |
| Amol Dhole | 71 | 224013 |

Under Guidance Of

**Mr. P.S.R. PATNAIK**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**WALCHAND INSTITUTE OF TECHNOLOGY**

**SOLAPUR - 413006**

**(2018-2019)**

****

**CERTIFICATE**

This is to certify that the Project entitled

**“​ ChatBot To Bargain Price For Woo-commerce Portal”**

**Is**

**Submitted by**

|  |  |  |
| --- | --- | --- |
| Name | Roll. No. | Exam Seat No. |
| Chayan Bobra | 09 | 807028 |
| Kartik Jain | 10 | 807129 |
| Kunal Burgul | 11 | 807046 |
| Sujay Khandare | 12 | 807110 |
| Amol Dhole | 71 | 224013 |

As a part of Project Design Report.

Studying in BE CSE for the subject **Lab II - Project Phase II**

**(Mr. P.S.R. PATNAIK) (Dr.R.V.Argiddi) (Dr. S.A. Halkude)**

***Project Guide* HeadPrincipal**

**Dept of Computer Science & Engg**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**WALCHAND INSTITUTE OF TECHNOLOGY**

**SOLAPUR**

**(2018-2019)**

**Plagiarism Undertaking**

We solemnly declare that project work presented in the report titled “…………………………………………………………………………………………….

ChatBot To Bargain Price For Woo-commerce Portal

…………………………………………………………………………………………….”

is solely our project work with no significant contribution from any other person. Small contribution/help wherever taken has been duly acknowledged and that complete report has been written by the members of the project group.

We understand the zero tolerance policy of the WIT, Solapur and University towards plagiarism. Therefore we as Authors of the above titled report declare that no portion of the report has been plagiarized and any material used as reference is properly referred / cited.

We undertake that if found guilty of any formal plagiarism in the above titled report even after award of the degree, WIT, Solapur and Solapur University reserves the rights to withdraw/revoke the degree granted and that WIT, Solapur and the University has the right to publish our name on the website on which names of students are placed who submitted plagiarized report.

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Student | Exam Number | University PRN Number | Signature |
| Chayan Bobra | 807028 | 2015032500254526 |  |
| Kartik Jain | 807129 | 2015032500253546 |  |
| Kunal Burgul | 807046 | 2015032500251266 |  |
| Sujay Khandare | 807010 | 2015032500253523 |  |
| Amol Dhole | 224013 | 2012032510003131 |  |

Date: / / .

**Acknowledgment**

At the outset, we would like to take this opportunity to express our deep gratitude to

our guide **Mr.P.S.R. Patnaik** of CSE Department for his guidance and moral support throughout our project.

We heartily thank **Prof. Mr.R.V.Argiddi**, Head of CSE Dept for his moral support and

promoting our project.

We would also like to thank our Principal **Dr. S. A. Halkude** and all staff members for

their whole hearted cooperation in working on this project.

**INDEX**

|  |  |  |
| --- | --- | --- |
| S.no | Topic | Page No. |
| 1 | Abstract | 06 |
| 2 | Introduction | 07 |
| 3 | Existing Systems | 11 |
| 4 | Methodology | 13 |
| 5 | Dependencies and Requirements | 21 |
| 6 | Instructions for deployment | 23 |
| 7 | Summary | 26 |
| 8 | Future scope | 27 |
| 9 | Reference | 28 |

**Abstract**

E-commerce is one of the extremely fast-paced marketing business around the world. Over the last few decades, several major e-commerce organizations have emerged such as Amazon, Flipkart that sell the majority varieties of merchandise and also other competitors like Dollar shave club and big basket are playing a major role in this high competitive business with domesticated products.

All of these businesses offer an internet catalog from that customers choose and purchase. However, in a more traditional setting, there's one key part that most E-Commerce platforms don't provide: Customer-Retailer interaction (Bargaining of Product). Due to the traditional human psychology of bargaining, a product for a lower price still being prominent also lack digital awareness being cause for some of the products in this business.

Advantage of the use of this e-commerce manner of marketing may be available in any respect time consistent with the benefit of the customer which can also be the motive of gaining some profit within the marketplace. However, for almost, all merchandise/products charges vary. E-Commerce is a method to make the buying experience for clients extra smooth and interactive with the aid of introducing bargaining features, which are otherwise handiest familiar in physical brick and mortar stores. ChatBot additionally also utilizes machine learning in order to sell prices for sellers to look at and make a more knowledgeable and proficient selling decision.

In This Project, The Customer-Retailer interaction is our goal to imitate online in a reasonable and can be scaled. This can also be designed in a way that could provide gains for retailers to attract the majority of the attention and also all offers made by retailers need to have a specific goal to be reached to sell. Through this Bargain Model, Customers get benefited with the aid of buying the product at its quality possible price.

# 

# 

# 

# 

# 

**Introduction**

The challenge of negotiation arises, in part, from the fact that each side has private information about their own utility function, but is ignorant of the other's values and strategies. Exacerbating this situation is the incentive that negotiators have to misrepresent their preferences[3]. Here negotiation is a communication process among parties having conflicting interests to reach an agreement. Online auction is a type of one-to-many negotiation and it is the main trading mechanism in the electronic market which proves that re-negotiation is going to play a major role in e-commerce.

Even in simple negotiations, people often reach sub-optimal negotiations thereby "leaving money on the table"[1][2].While many factors contribute to missing out on gains from trade—overconfidence, falsely assuming fixed price, and the framing of the situation the end result is that parties fail to find agreements which would make each better off[3].

From the beginning when online retailing began, a lot of sites and tools came to help the shoppers so that they can find the available best deal (i.e. the lowest price). These are called “shopping bots”. These shopping bots are the website or apps which search the other sites for the same product and compare the prices and help buyer to find out the best available price by showing him the comparative pricing chart. But none of the sites really offers the bargaining power to the shopper.

This well-documented fact has led researchers to develop tools to help people prepare for and participate in negotiations. This project looks toward future electronic marketplaces and investigates not just supporting negotiators but also the possibility of fully automated business negotiations.

Our system aims is to give a maximum discount to users using factors like product id, customer id, and quantity to bargain the price of the product and also uses this data to train an ML (Machine Learning) model, to predict future prices of products.

Product id and customer id will be generated by the site and quantity in which user want to buy will be given by the user to the site.

The concept of Online Bargain describes the bargaining process that undergoes customer interaction with the online website for some transactions or purchases. This process will set a different level of price for different products, on the basis of an analysis of frequency and product sold.

The price of a particular product will be set for a particular day as per the result of analysis of that product quantity, Product ID & customer ID. Customer enters the bargain amount for a particular product, that bargaining amount is compared with the selling price of that product. In this case, the result will be either of two cases. Firstly, the product will be a deal if the amount entered is within the context of the deal. If not, Customer will be asked to bargain again for the second attempt. In either case, the customer is free to deal anytime or exit the process.

Maximum of three attempts are provided so as to bargain and deal on that product. So different customer will get a benefit of a different amount, It is similar to the customer of a shop wherein different individual will bargain for a different amount. Each transaction here is considered as a separate thread which is totally independent of other thread.

This concept is implemented keeping the benefit of the online retailers in mind. The database

maintained for this purpose extracts data from different websites, on an hourly basis, from which the products are dealt with[4].

In this process, we have to keep a track on some issues like[4]:

* Collection of the data of each product.
* Dealing with different websites for different product data.
* Analysis of profit amount to be added to the actual amount.
* No. of bargaining attempts available for the individual.
* Comparison of the bargained amount with product price

.

We tried to emulate customer-retailer interaction at a basic level by introducing a bargaining feature, which allows customers and vendors to agree to a new price at some rudimentary level.

There is a lot of scope of bargaining in the E-Commerce sites, but it is not possible for the e-commerce companies to hire so many persons as it will be impractical and impossible to have one person per shopper. Also, it will be very costly to have so much staff strength because their expenditure to maintain large staff will be more than their profit.

Therefore surely some machine (software) is needed which can do bargaining on behalf of the seller.

not only for all online consumers who are actively buying the product online but it also made for that consumer who is not actively engaged on e-commerce portal. This system helps those people by giving a fair amount of discount price based on their activities so it tempts

to engage more and more.

***2.1 Factors :***

2.1.1) Product Id:

Its unique id which is given to all products on site. This id will be required to identify categories of that particular product on site.

2.1.2) Customer id:

Its unique id which is given to all existing customers which are registers on site. This id will be required to fetch the detail of that customer like a customer with id 11230 is actively buying products from the site and it will also give detail of that product category.

2.1.3) Quantity:

It is the quantity of product which customer is ready to buy.

***2.2 Front End:***

2.2.1) HTML :

HTML[6] is the standard markup language for creating a web page and web applications. With Cascading Style Sheets (CSS) and JavaScript.By using HTML and CSS we have created the front end of our project. A based document may be created with the help of structural semantic for textual content like heading, paragraph, list, link, and different gadgets. Browser indeed does not display the HTML tags but utilize them to interpret the content of the page.

2.2.2) CSS:

We have used CSS[6] for styling of HTML Web pages, including colors, layout, and fonts. It allows one to adapt the presentation to different types of CSS is independent of HTML.

***2.3 Back End***

2.3.1) Flask:

Flask[5] is a lightweight web application framework written in python and baseband on the WSGI toolkit and jinja2 template engine. Flask takes the flexible python language and provides a simple template for web development. Flask can be used to save time building web applications after imported into python. It has no database abstraction layer, form validation, or any other components. It keeps the core simple but extensible.

Feature of flask:

* Integrated supports for unit testing.
* Uses Jinja2 templating
* Support for secure cookies
* Extensive documentation
* Google app engine compatibility
* Restful request dispatching
* Unicode based

2.3.2) Python:

There is no type of declaration of variables, parameters, functions, or methods in the source code. This makes the code short and versatile, and you lose the compile-time sort. It tracks the types of all values at run time and flags code that does not make sense as it runs. It is the most trending powerful and fast, runs everywhere is friendly and easy to learn. The ".py" extension is used for Python source files are called "modules”.

2.3.3) Pandas:

Pandas is a Python package offering fast, flexible, and expressive information structures designed to make operating with “relational” or “categorized” facts each clean and intuitive.

Python has long been amazing for data munging and preparation, but much less so for data analysis and modeling. Pandas enable fill this hole, permitting you to carry out your complete information analysis workflow in Python without having to switch to a more domain specific language.

Pandas is nicely acceptable for many different sorts of information:

• Tabular facts with heterogeneously-typed columns, as in an SQL desk or Excel spreadsheet

• Ordered and unordered (no longer necessarily fixed-frequency) time collection records.

• Arbitrary matrix records (homogeneously typed or heterogeneous) with row and column labels

• Any different shape of observational/statistical information units.

2.3.4) Numpy:

Numpy[7] is the most fundamental and effective package deal for running with facts python. At the core, numpy gives extraordinary nd-array items, brief n-dimensional arrays. In a 'ndarray' object ‘array’, you may keep multiple objects of the identical information kind. It is the facilities around array item that makes numpy so handy for acting math and facts manipulations.NumPy array is a multidimensional, uniform collection of elements. An array is characterized through the form of factors and with the aid of its shape.

2.3.5) Scikit-learn:

Scikit-learn[8] is this type of Python module which integrating an extensive range of contemporary machine gaining knowledge of algorithms for medium-scale supervised and unsupervised issues. This package deal makes a specialty of bringing ML to non-specialists using a high degree language. Emphasis is placed on performance, the way it is easy to use, documentation, and API consistency. It has minimal dependencies and is sent beneath the simplified BSD license, encouraging its use in each educational and commercial settings.

Scikit-research differs from other machine learning toolboxes in Python for numerous reasons as an instance:

i) underneath the BSD license Scikit-examine is distributed.

ii) it consists of compiled code for more efficiency.

2.3.6) Pickle:

Python’s built-in Pickle[9] module implements an algorithm for serializing and deserializing objects, commonly for persistence or transport. Python Pickle module provides a known capability for running arbitrary Python functions and, by extension, permitting remote code execution; however, there is no public Pickle exploitation guide and published exploits are simple examples only. In this paper we describe the Pickle environment, outline hurdles facing a shellcode and provide guidelines for writing Pickle shellcode. A brief survey of public Python code was undertaken to establish the prevalence of the vulnerability, and a shellcode generator and Pickle mangler were written. The Output from the paper includes helpful guidelines and templates for shellcode writing, tools for Pickle hacking and a shellcode library.

**Existing Systems**

In the current existing eCommerce system, offers are initiated from the vendor's side. Vendors offer prices on products at rates at either MRP or lower in times of sale. There is no provision for customers to buy in mass, i.e customers are serviced individually rather than in bulk. So vendors sell products usually at slower single unit rates instead of bulk unit rates. Also in current eCommerce sites, there is no dynamic of the customer being able to bargain for a lower rate. This reduces the power of the customer and the flexibility of the customer.

Various researches have been done in this area of dynamic pricing and online bargaining, which include work done by Fu-Ming Lee, Li-Hua Li and Pao-Hsiao Chen[10], which overcomes the time constraints issue, so as the strategies of the offers and reservation price is not revealed to the customer[4].

Paper on a genetic algorithm based bargaining by Kumar Ujjwal and Jay Aronson[11] has signified that change of price of a product is superior to the fixed product price. As to benefit the retailer and make the customer feel that he is also benefited, there are different ways of dynamic pricing, The work presented in this paper purposes simple and elegant way to implement online bargaining using genetic algorithm and mutation operators[4].

Facebook’s FAIR[12] researchers allowed the model to achieve the goals of the negotiation. To train the model to attain its goals, the researchers had the model apply thousands of negotiations against itself, and used reinforcement learning to reward the model once it achieved an honest outcome. To prevent the algorithm from developing its own language, it was simultaneously trained to produce human-like language. To evaluate the negotiation agents, FAIR tested them online in conversations with people. Most previous work has avoided dialogs with real people or worked in less challenging domains, because of the difficulties of learning models that can respond to the variety of language that people can say.

Interestingly, in the FAIR experiments, most people did not realize they were talking to a bot rather than another person — showing that the bots had learned to hold fluent conversations in English in this domain. The performance of FAIR’s best negotiation agent, which makes use of reinforcement learning and dialog rollouts, matched that of human negotiators. It achieved better deals about as often as worse deals, demonstrating that FAIR’s bots not only can speak English but also think intelligently about what to say[12].

Through this Project we tried to shows that the people might prefer shopping online at e-stores that offer the chance to negotiate a price. This framework has a series of the shift in the price that will be dealt if both sides agree on the price. The Product Negotiation tries to emulate customer-retailer interaction at a basic level by introducing a bargaining feature, which allows customers and vendors to agree to a new price at some rudimentary level.

Background Business entities have tried for years to adapt computers and networks for use in sophisticated intercompany negotiations for commercial purchase and sales transactions, but with results that usually fall far short of expectations. Early mainframe computer attempts, for example, usually involved one corporation's allowing its existing suppliers and quantity buyers to connect to its internal private, proprietary network, using specially written locally developed application programs and private, proprietary network connections. These private systems were usually extremely costly to develop and maintain (often costing in the multi-millions of dollars) and very often did not meet all the needs and changing requirements of the participating businesses. Since many corporations had different internal networks and computer systems, considerable effort went into working around incompatibilities. Additionally, these systems had to be based on already existing, close relationships between buyers and sellers and usually were also based on previously negotiated agreements. Thus, the systems did not help in searching for information about new buyers and sellers, nor with the evaluation or negotiation processes, nor with the documenting of those processes from the beginning. They were not interactive, but typically batch processing systems, and usually accepted alphanumeric text only, not the inclusion of graphics or sound files. They usually addressed ongoing relationships previously worked out manually, for which extremely expensive custom systems were developed at buyers' or vendors sites.

Most business (and many other) negotiation processes are usually multivariate. That is, a business negotiation deals with many variable items, such as price, quantity, quality, shippers, insurance, warranty, schedules, returns and so on. The above solutions typically did not automate multivariate negotiations in any way, since they had to be built on agreements whose terms had all been previously negotiated

# 

# 

**Methodology**

***4.1 System Architecture:*** It conceptual model that defines the structure, conduct, and extra views of a gadget. An architecture description is a formal description of a device, prepared in a manner that helps to reason about the systems and behaviors of the system.

***4.2 Train the model:*** The first step is to train the model based on the use case using Machine Learning Algorithms. Make sure we do this in a virtual environment, as it helps in isolating multiple Python environments and also it packs all the necessary dependencies into a separate folder.

***4.3 Build the API:*** Once the model is good to go into an API, we can use FLASK to build them based on the requirement. Ideally, we have to build Restful APIs, since it helps in separating between the client and the server; improves visibility, reliability, and scalability; it is platform agnostic. Perform a thorough test to ensure the model responds with the correct predictions from the API.

***4.4 Web Server:*** Now is the time to test the web server for the API that we have built.

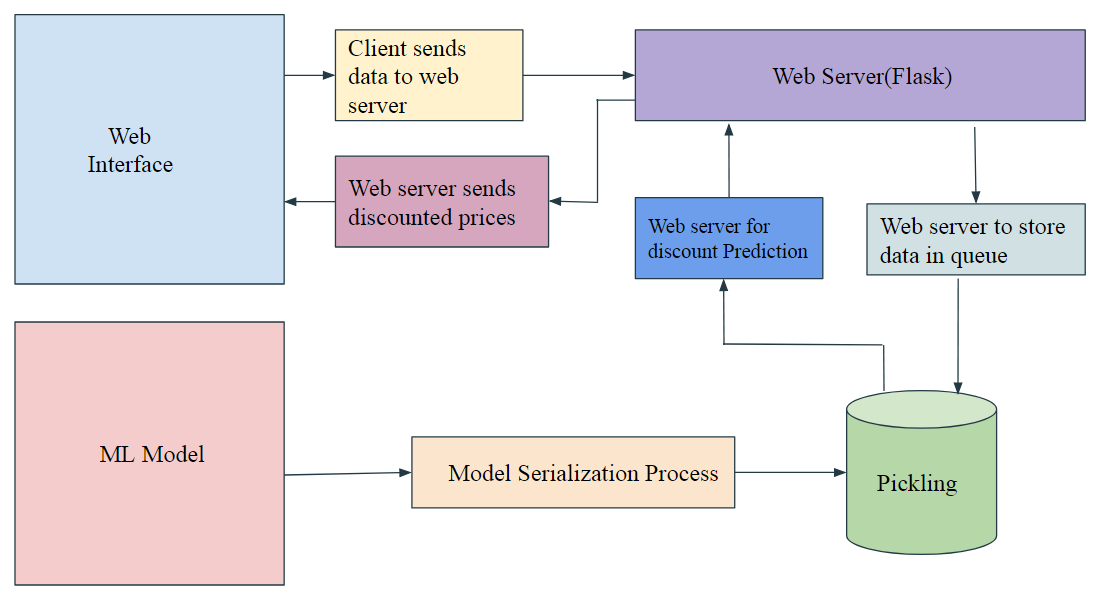
****

Fig 4.1: System Architecture

4.5 ***Machine Learning Model:***

4.5.1) Data Exploration:

As there are few datasets available within the marketplace associated with the Ecommerce which incorporates the facts related to the customers and products. So we can be relying on the Brazilian E-Commerce Public Dataset by Olist[13] from kaggle and the Online Retailer dataset from the UCI Machine learning repository[14]. As most of the wished feature found inside the Online Retailer dataset, so we are using this dataset to train our ML model.

4.5.2) Data Preprocessing:

We are using the Online Retailer Dataset to Train the ML model, so right here are the following features available in this dataset[14].

* InvoiceNo: It is a unique number assigned during each transaction.
* Stock Code: It is for distinguishing a product unique and it is a 5 digit number.
* Description: Contains details of the product and various features.
* Quantity: Indicates the quantity available and can be availed for each transaction.
* InvoiceDate: It indicates the exact time and date during which the transaction happens.
* Unit Price: It indicates pricing per unit.
* CustomerID: A unique number assigned to each customer who is visiting the site.
* Country: It gives information which country customer belongs to.

As our principal goal to have a negotiation with the customer we want the ML version too is expecting the discount for that purchaser based at the info consisting of CustomerID, Stock Code as ProductID, Quantity the customer is buying and the UnitPrice. These are the required features that we are using for training our ML model.

Now we are developing a new feature Discount for our dataset from the existing feature UnitPrice.

Here is the formula,

**MRP = ((max(UP )(i))\* 1.1)**

**Discount(i)  = ((MRP – (UP)(i)) / MRP)\*100**

UP - Unit Price

MRP - Maximum Retail Price

for ex:

Normal Features of the dataset that are considered

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr no | ProductID | CustomerID | Quantity | UnitPrice |
| 1 | 10080 | 14535 | 1 | 0.39 |
| 2 | 10080 | 16712 | 1 | 0.39 |
| 3 | 10080 | 17870 | 2 | 0.39 |
| 4 | 10080 | 15547 | 2 | 0.85 |
| 5 | 10080 | 18096 | 3 | 0.39 |
| 6 | 10080 | 16924 | 4 | 0.39 |

Now, calculating Discount

MRP = (0.85 \* 1.1)

MRP = 0.935

Discount(1) = ((0.935 - 0.39) / 0.935) \* 100

Discount(1)= 58.28

Features of dataset when Discount is taken in consideration

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr no | ProductID | CustomerID | Quantity | UnitPrice | Discount |
| 1 | 10080 | 14535 | 1 | 0.39 | 58.28 |
| 2 | 10080 | 16712 | 1 | 0.39 | 58.28 |
| 3 | 10080 | 17870 | 2 | 0.39 | 58.28 |
| 4 | 10080 | 15547 | 2 | 0.85 | 9.09 |
| 5 | 10080 | 18096 | 3 | 0.39 | 58.28 |
| 6 | 10080 | 16924 | 4 | 0.39 | 58.28 |

So here is the final dataset that we are using with the following feature ProductID, CustomerID, Quantity, UnitPrice and Discount.

4.5.3) Model Implementation:

The implementation process can be split data into training data and testing data. As we are having a good amount of data nearly 100143 so we will be using the 75% as the training data and the remaining 25% as the testing data as shown in the below figure

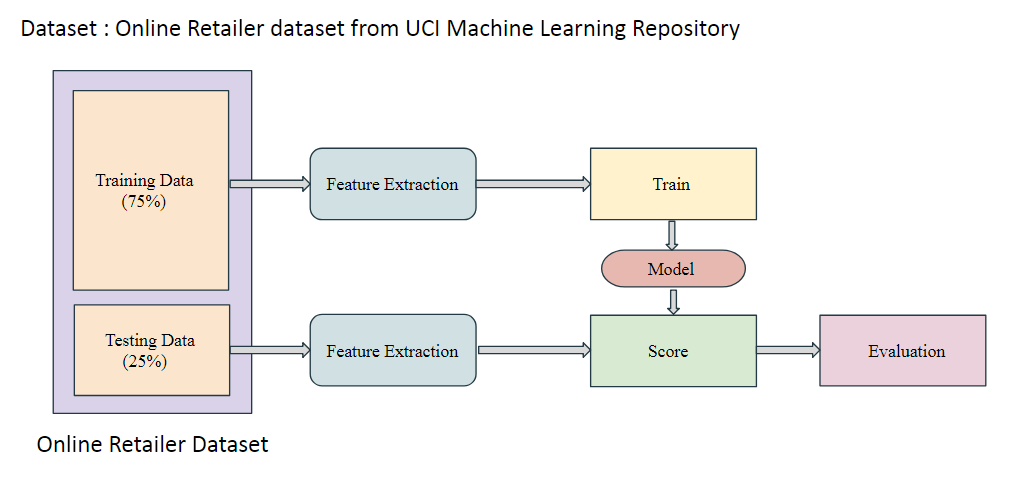
****

Fig 4.2 : ML Model Overview

As our dataset is absolutely set now we are able to pass further to apply the Machine Learning Algorithms.

Here ProductID, CustomerID, and Quantity are few of the independent variables and Discount is the dependent variable for the regression model. As there are a couple of independent variables so first, we’ve attempted with the Simple Multiple Linear Regression as the dataset was not fitting properly so we have decided to move for a different algorithm.

As the observed results were appropriate after implementing through decision tree regression model so we have taken this algorithm into consideration. A decision tree is a type of supervised learning algorithm that is mostly used in classification problems. It works for both continuous and categorical input and output variables.

In Decision Tree, we break up the data or sample into two or greater homogeneous units (or sub-populations) based on maximum widespread splitter/differentiator in input variables m.



Fig 4.3: Decision Tree

We have likewise attempted to actualize with the Random Forest and XGBoost to achieve more accuracy. Using Random Forest we achieved accuracy closer to the decision tree but as we were using some basic tuning parameters so this was time-consuming. As this dataset was not working properly XGBoost so we decided to stick to the Decision Tree Regression.

4.5.4) Accuracy

The key features/attributes that are passed as in input to the ML model are ProductID, CustomerID, and Quantity. We are predicting Discount as an output from the ML model. The accuracy (r2\_score) so far we have achieved for this Decision Tree ML model is 85.54% as shown in the below figure.

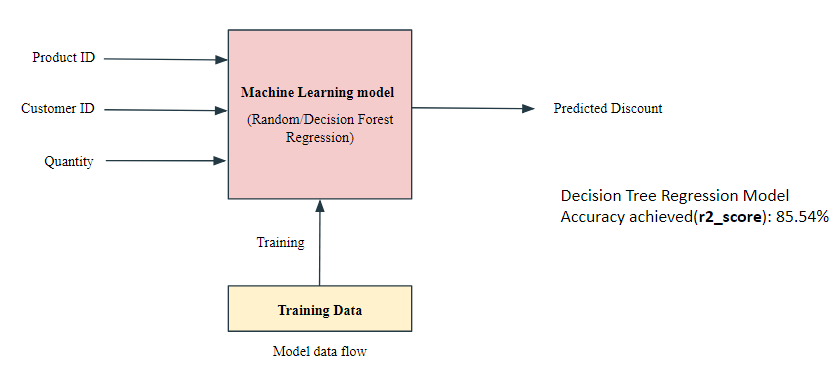


Fig 4.4: Predicted Discount Model with accuracy Score

4.5.5) Generating a Pickle File

Once the ML model is finalized with good accuracy results. Then only one time the training is done and we generate a pickle file with .pkl extension using the following code:

pickle.dump(regressor, open("model.pkl","wb"))

Here in the above code, Pickle is used to serializing and de-serializing a Python object structure. In this python, the object is converted into the byte stream where dump() method dumps the object into the file specified in the arguments. Now, this model.pkl file can be used as the ML model were the input tuples are passed to the model.pkl file and the result is predicated. This is the process of how the pickle file is used as an ML model.

4.5.6) API Connectivity

We will use Flask it is a very light web framework. We created \_\_init\_\_.py flask file. This acts as an API interface to connect the front-end Web and the backend ML model.

Here all the input parameter that is to be passed to the ML model are retrieved from the web using the GET method from the \_\_init\_\_.py[15].

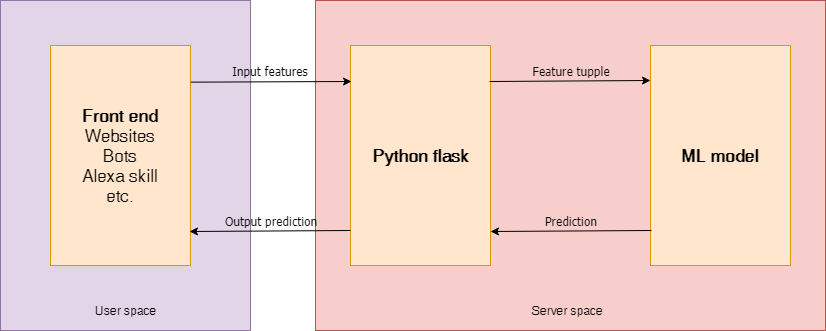
****

Fig 4.5: Python Flask Rest API Integration

GET: Browser requests server to get data in an unencrypted form stored on that page and send it. It’s the default method. It can be cached and remains in browser history.

All this retrieved data is sent to the model.pkl file to predict the Discount. These input parameters are ProductID, CustomerID, and Quantity. Once this process is completed then the predicted Discount is taken into consideration,

loaded\_model = pickle.load(open("model.pkl", "rb"))

result = loaded\_model.predict(to\_predict)

The above code is used to load the input parameters and generate the prediction and store in the result. Here the POST method is used to pass back the result to web/frontend from the flask file \_\_init\_\_.py. In this way, the results are reflected in frontend.

POST: It is used to send HTML form data to be processed to a specified resource. The browser tells the server that it wants to posts some new data to the URL and it is only stored once. Here the data received by the POST method is not cached by the server.

GET and POST these are some of the important methods used repeatedly in the \_\_init\_\_.py flask file.

Before passing the results to the frontend here is the main negotiation parameter is decided. As we are allowing the customer to bargain at most 3 time so here is the following process followed. The predicted discount is used here in this

This three-time bargaining process is as follows:

At the first time, 50% of the predicted Discount is taken into consideration

disprice = (int(actual price) \* ((result/100)/2))

The second time, 75% of the predicted Discount is considered

disprice = (int(actual price) \* ((result / 100)\*(3/4)))

Finally the whole predicted result is considered

disprice = (int(actual price) \* (result / 100))

In this way, the bargaining process works this is how the predicted values Discount will help us to negotiate with the customer as this will also help from losing the customer and increasing the productivity to the seller[8].

Once this disprice is finalized the code for the final price is as follows

finalprice = (int(actual price) - disprice)

There may arise a situation where the final price may be less than the user price, in that case, the below code will be executed

if float(finalprice) < float(userprice):

finalprice = userprice

4.6 Frontend Website / Output

The Frontend of the website is an HTML webpage with CSS styling and javascript for the client side scripting. All of the .html files are stored in the templates folder and other images, scripting, and .css files are stored in the script folder. Here the user enters the values in <input> tags which are retrieved by the Flask API and then processed by the ML model the results are sent back to the Website using the same Flask framework.

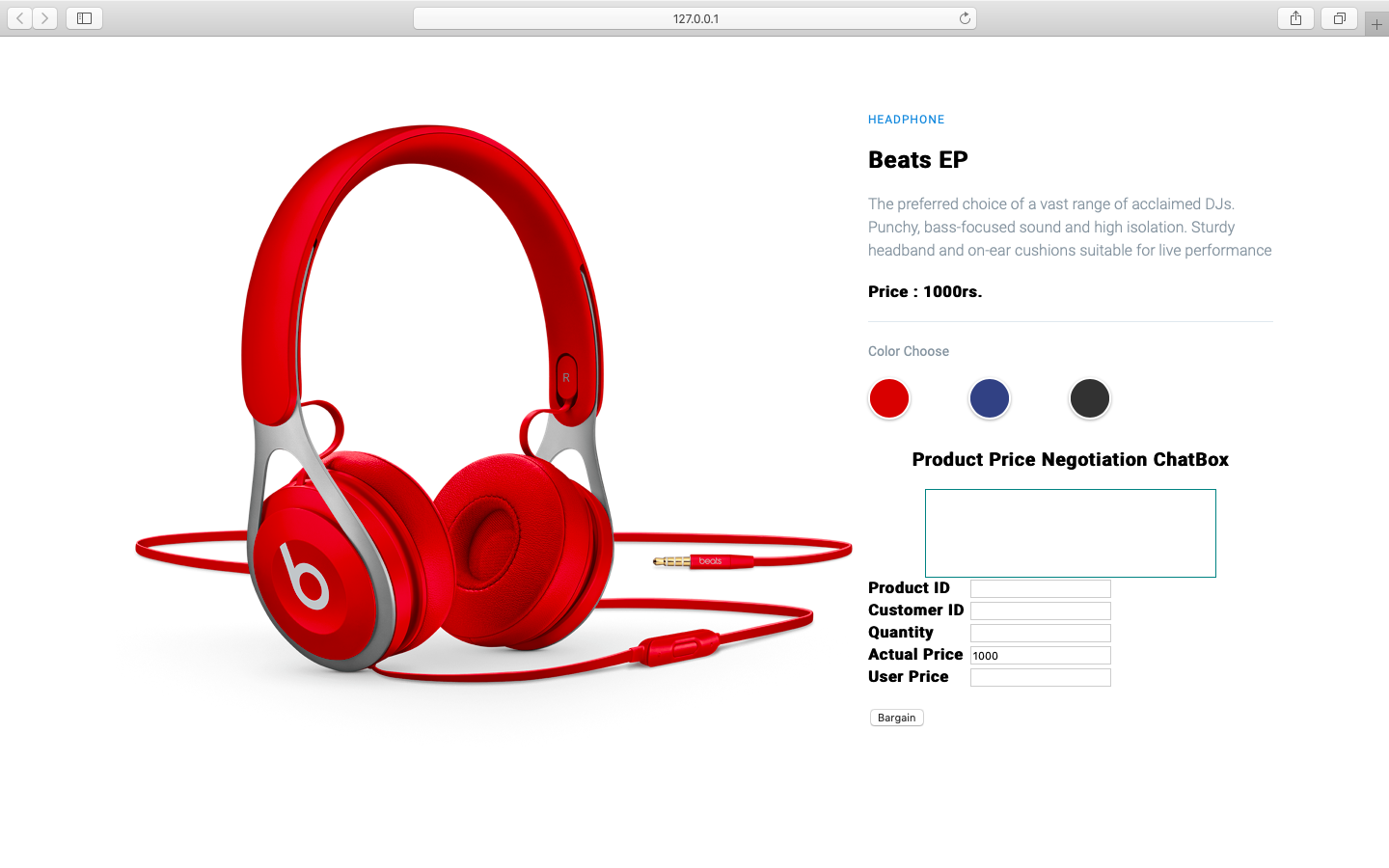


Fig 4.6 Frontend view

# Dependencies and Requirements

# 5.1 *Libraries used*:

5.1.1) flask 1.0.2

Flask is a lightweight WGSI(Web Server Gateway Interface) web application framework. It began as a simple wrapper around werkzeug and jinja and has become one of the most popular Python web application frameworks

License: BSD Licence

5.1.2) cloudpickle 0.8.1

Cloudpickle 0.8.1 is used for serializing and de-serializing

License: BSD 3-Clause

5.1.3) jinja 2.10.1

Jinja is web template engine for the python programming language

License: BSD Licence

5.1.3) numpy 1.16.2

A fundamental package for array computing in python

License: OSI Approved (BSD)

5.1.4) pandas 0.24.2

Pandas is a open source, easy to use data structures and data analysis tools for Python programming language

License: BSD Licence

5.1.5) scipy 1.2.1

Scipy is a open source python library used for the scientific computing and technical computing

License: BSD 3-Clause

5.1.6) scikit-learn 0.20.3

Scikit-learn is a free software machine learning library for python programming.

License: BSD 3-Clause

5.1.7) werkzeug 0.15.2

Flask wraps Werkzeug, using it to handle the details of WSGI while providing more structure and patterns for defining powerful applications.

License: BSD Licence

***5.2 Software used:***

5.2.1) Python 3.6 or higher

5.2.2) Anaconda 3 or higher

***5.3 IDE’s Used:***

5.3.1)PyCharm Community Edition for Flask API

5.3.2) Spyder for the ML model

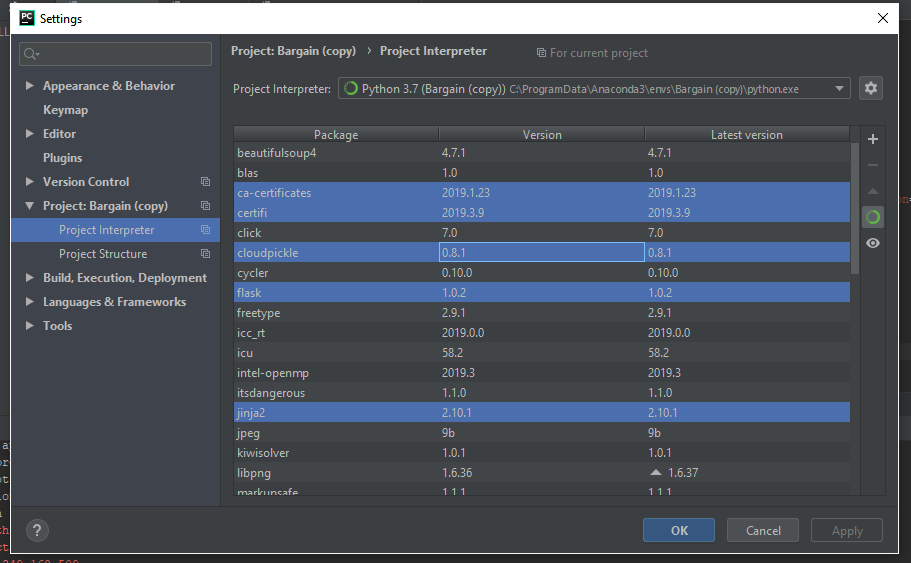
**Instruction for Deployment**

Step 1 : Download and install the basic requirements and the IDE.

Step 2 : Once the basic requirement and IDE’s satisfied then install all the libraries from

dependencies and requirements.

For easy installation you can do this through PyCharm Interpreter.



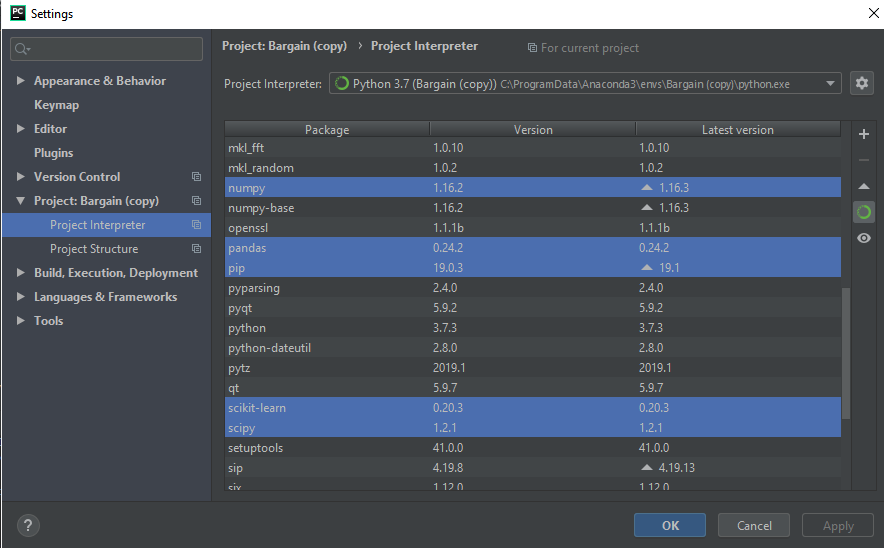
****

Fig 6.1 PyCharm Interpreter

Step 3: To check accuracy of the ML model you can run the DecisionTreeRegression.py file

present in ML Model folder from Spyder through Anaconda Navigator.

You can get the accuracy of the Regression model by r2\_score in the code.

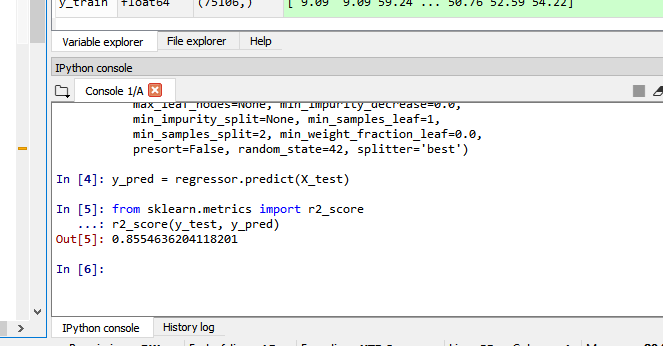


Fig 6.2: Accuracy Score

Step 4: Now to execute the project you need to run the \_\_init\_\_.py flask file present in the

main folder here when you run a flask file a localhost server is created by clicking on

the link you are directed to the website.

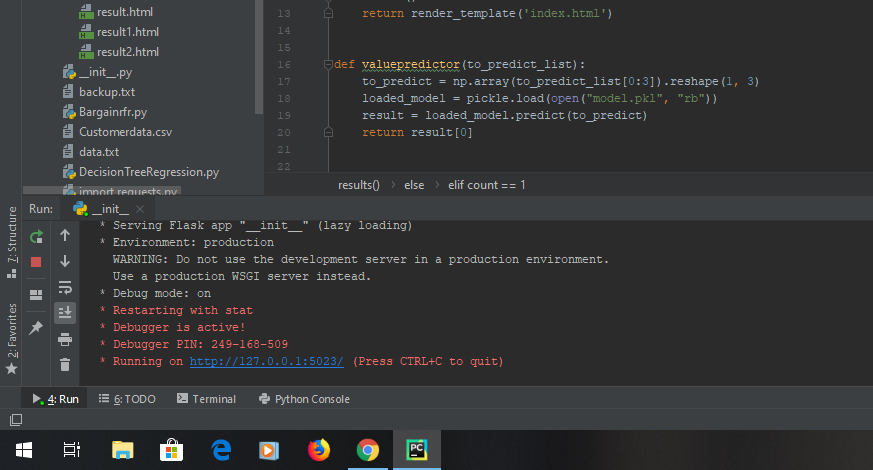


Fig 6.3 Execution of Flask file

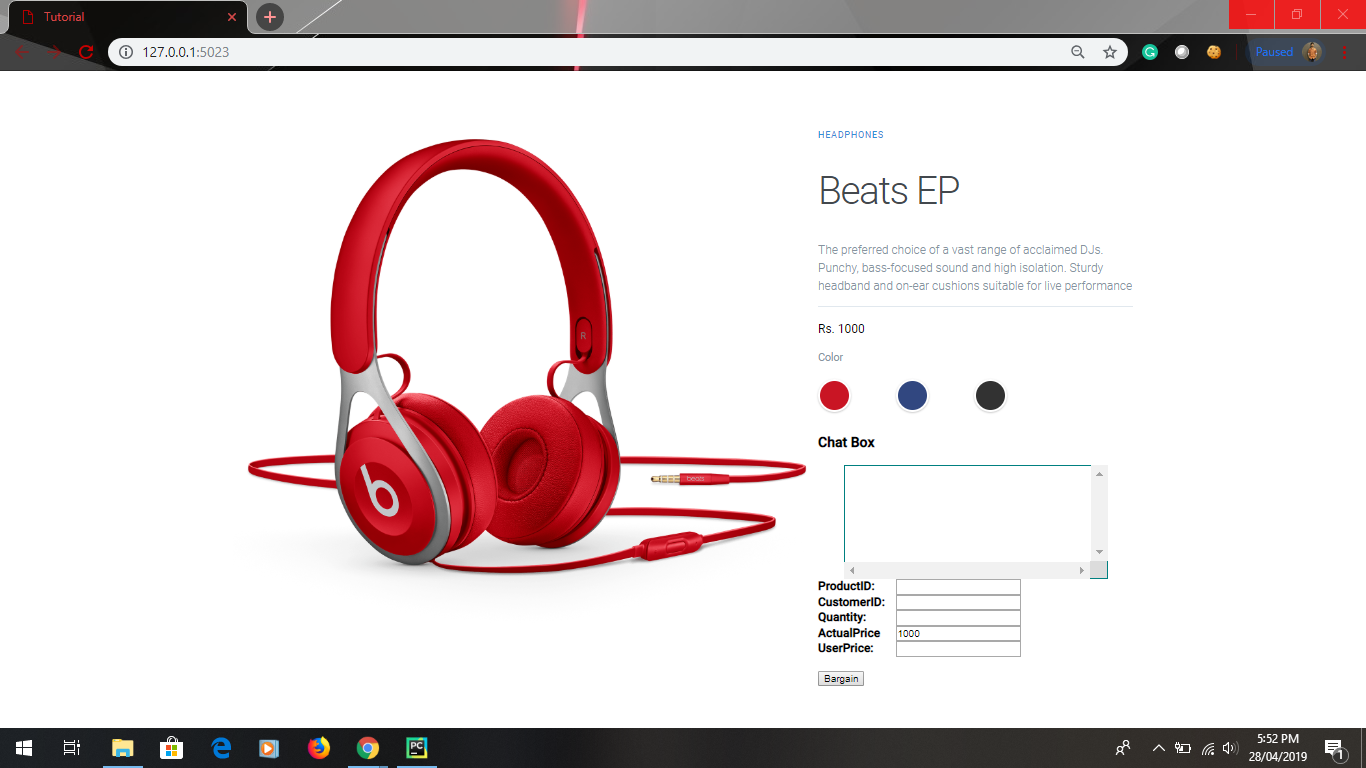


Fig 6.4: Frontend before negotiation

Step 5:Enter the ProductID, CustomerID, Quantity and UserPrice

Userprice - Userprice is a price a which Customer expecting to buy the product

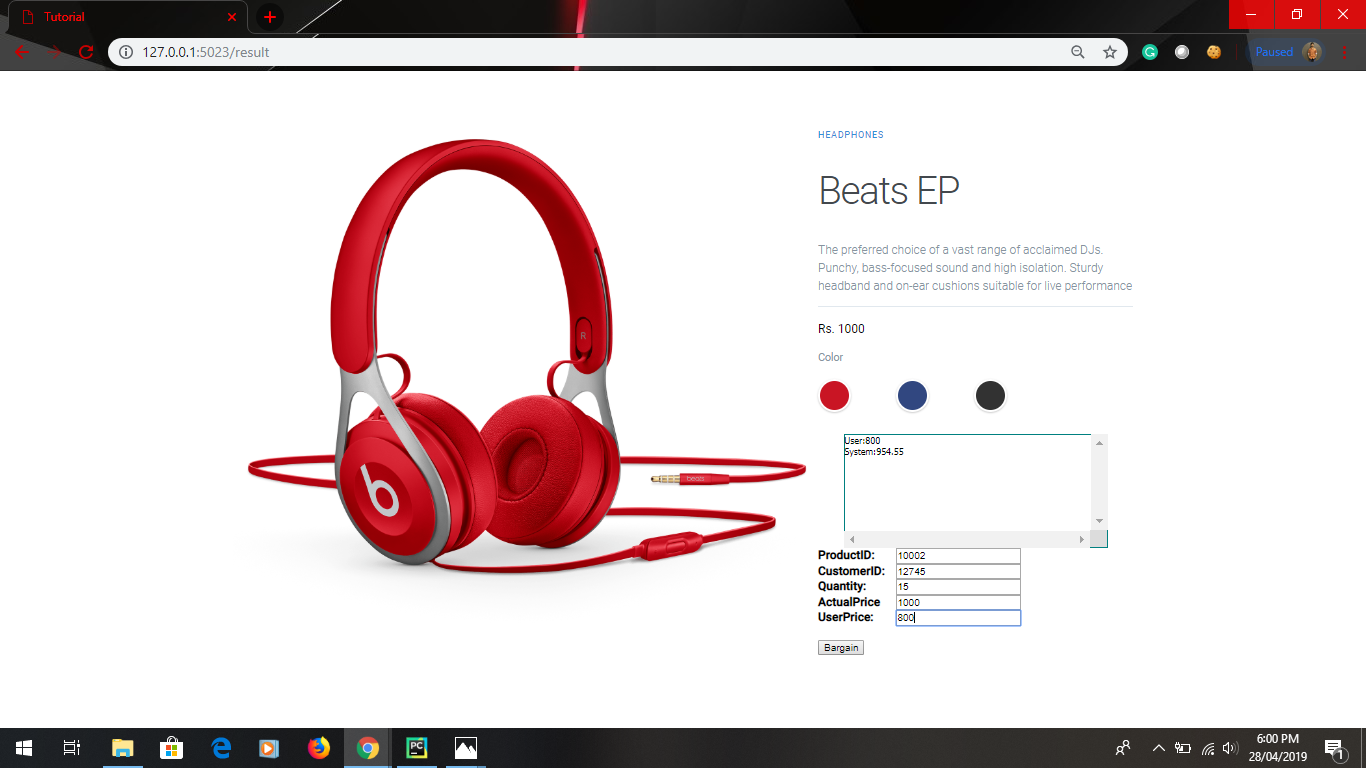


Fig 6.5 Frontend during negotiation

**Summary**

Exploring through Kaggle and UCI Machine Learning repository there were only few datasets which were fitting to this problem statement these were Brazilian E-Commerce Public Dataset by Olist from kaggle and the Online Retailer Dataset from the UCI Machine learning repository. As the Online Retailer Dataset was satisfying all the required features with massive data compared to Brazilian E-Commerce Public Dataset. So, this dataset was taken in consideration for this project.

According to continuity of the data in the dataset different regression ML algorithms such as Decision Tree Regression, Random Forest Regression and XGBoost Regression were executed.

An approach was tried were XGBoost Regressor was not properly fitting to the dataset as that of the Decision Tree Regressor and Random Forest Regressor.So the Decision Tree Regressor with a good accuracy (r2\_score: 85.54%) was considered.

Flask is used as an API Connectivity for ML model and the Web Interface. Where user enters the inputs in web interface and these inputs are passed to the ML model with the help of flask web application framework written in python. Once the result are predicted then this result is processed and passed back to the web interface with the same framework. This result was to be displayed in a text area in the web interface line-by-line one above the other. But as there was issue with a jinja2 for writing to new line this part was not implemented.

This project is implemented by creating a website on Wordpress. Here Woocommerce plugins are needed in Wordpress to call API to fetch data. Based on the API for those factors which are required for bargaining; insomnia client is installed to extract those API’s and get the required data. Further chat interface with display on client side is used for sending price the in demand of particular product to trained model. There was uncertainty with Inbuilt plugin in Woocommerce for connecting WooBot with model. The cause for connectivity problem is due to unavailability of particular API in this context for connecting the Chat interface with the model and also is cause for failure. Following up on this alternatively a website is created using HTML & CSS which was successfully implemented.

**Future Scope**

In future we will work on audio based model so that user do not have to type their details and it will be create more user friendly environment. We will also work on our accuracy algorithm so that user can get maximum discounts from our system.

Closed domain generative model based chatbot can be created which is not only used for the price negotiation but also for the other activities on the ecommerce site.

Natural Language Processing (NLP) can be introduced for better understanding of human to chatbot

Deep Learning technique like ANN(Artificial Neural Network) can be introduced instead of Machine Learning for much better accuracy of the model

# 

# References

[1]Camerer, C., Behavioral Game Theory, in Insights in Decision Making: A Tribute to Hillel J. Einhorn, R.M.Hogarth, Editor^Editors. 1990, Univ. of Chicago Press: Chicago, IL. p. 311-336.

[2]Raiffa, H., The Art and Science of Negotiation. 1982, Cambridge, MA: Harvard University Press.

[3]Jim R.Oliver ,“A Machine Learner Approach to Automated Negotiation and Prospects for Electronics Commerce”

[4]Naveen D. Chandavarkar,“Dynamic Pricing with Historical Database Analysis For Online Bargaining”

[5]Fankar Armash Aslam ,Hawa Nabeel Mohammed Efficient ,”Way Of Web Development Using Python And Flask”.

[6]<https://www.w3.org/standards/webdesign/htmlcss>

[7]Stéfan van der Walt, Stellenbosch University South Africa,S. Chris Colbert, Enthought USA,Gael Varoquaux, INRIA Saclay France ,“The NumPy array: a structure for eﬃcient numerical computation”

[8]Fabian Pedregosa , Gael Varoquaux , Alexandre Gramfort , Vincent Michel , Bertrand Thirion “Scikit-learn: Machine Learning in Python”.

[9]Marco Slaviero,’’Sour Pickles Shellcoding in Python’s serialisation format”

[10]Fu-Ming Lee, Li-Hua Li and Pao-Hsiao Chen, “A Study on Dynamic Bargaining Strategy under Time Constraints and with Incomplete Information”,Proceedings of the 2005 IEEE/WIC/ACM

International Conference on Intelligent Agent Technology (IAT’05), 2005.

[11]Kumar Ujjwal, and Jay Aronson, “Genetic Algorithm based bargaining agent for Implementing Dynamic Pricing on Internet”, Proceedings of the 2007 IEEE Symposium on Foundations of Computational Intelligence,2007.

[12]Mike Lewis, Denis Yarats, Yann N. Dauphin, Devi Parikh, “Deal or No Deal? End-to-End Learning for Negotiation Dialogues”

[13]“Brazilian E-Commerce Public Dataset by Olist”, Kaggle

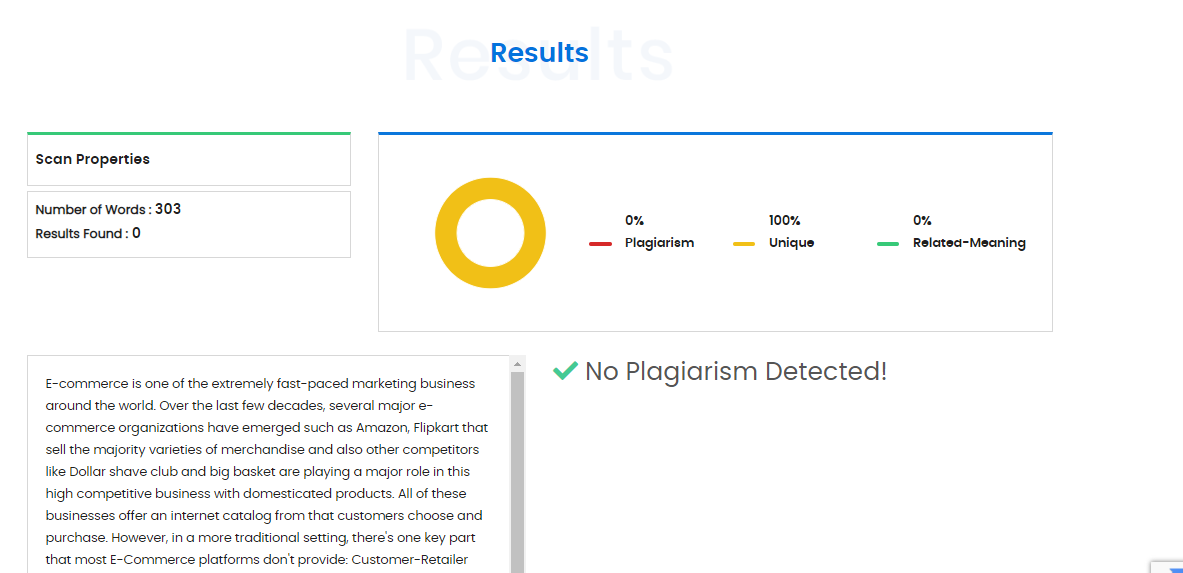
[14]“Online Retailer Dataset”, UCI Machine Learning Repository Center for Machine Learning and Intelligent Systems

[15] Vaibhav Kumar, “Deploy Machine Learning Models for Free Using Flask API from Medium”

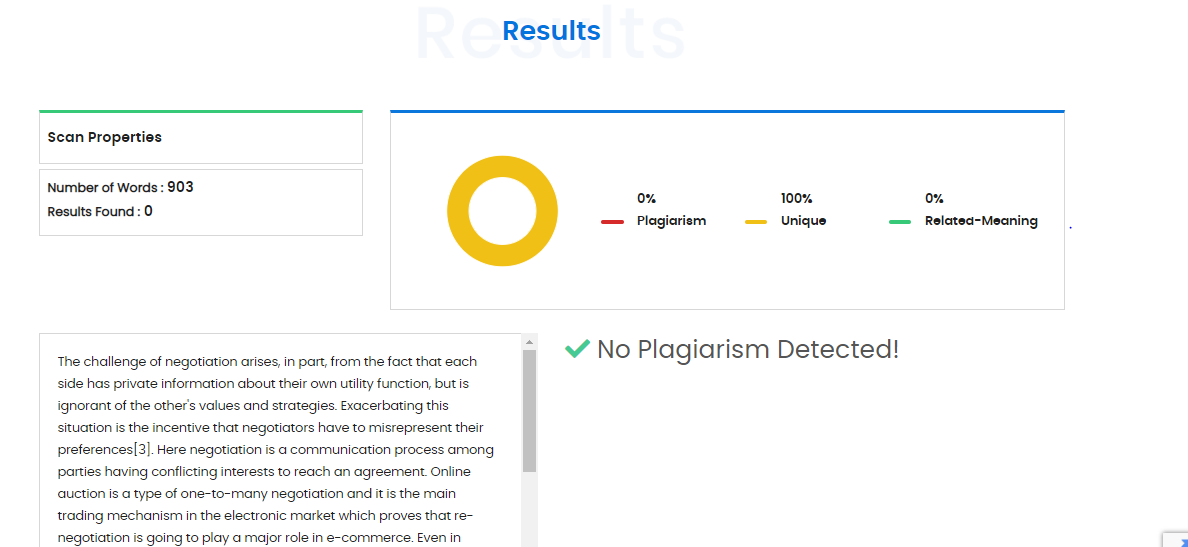
**Plagiarism check**

**Source :** https://www.duplichecker.com/

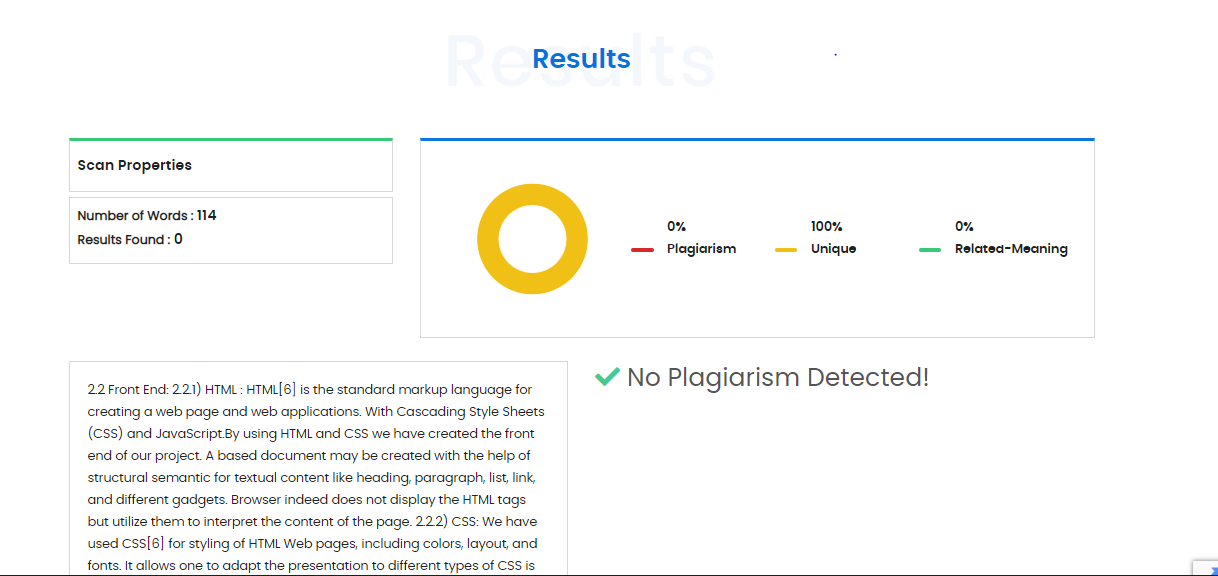
***Abstract:***

****

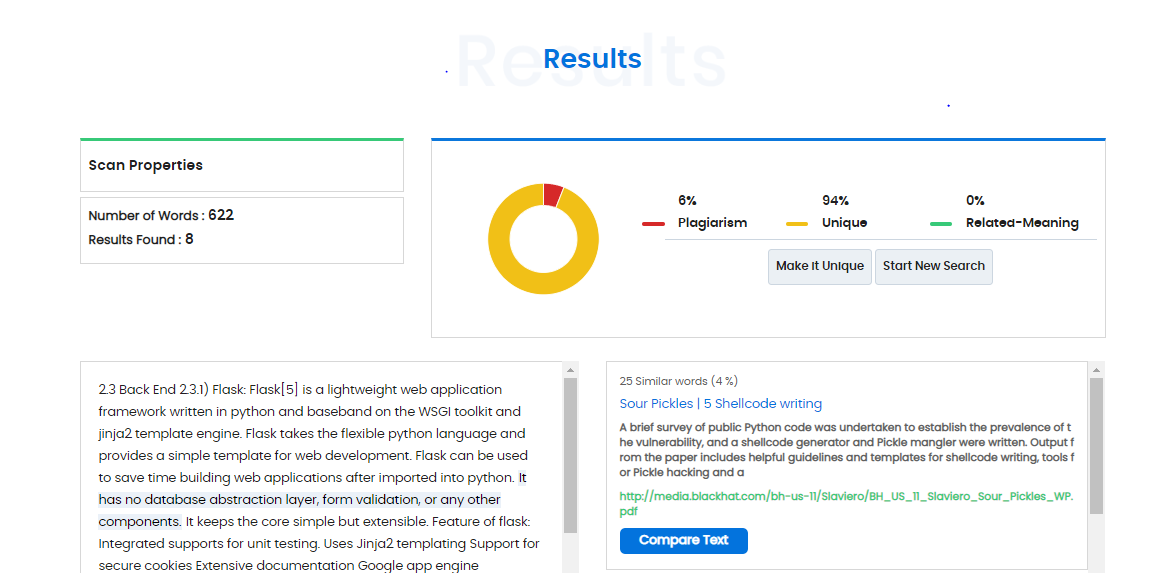
***Introduction***



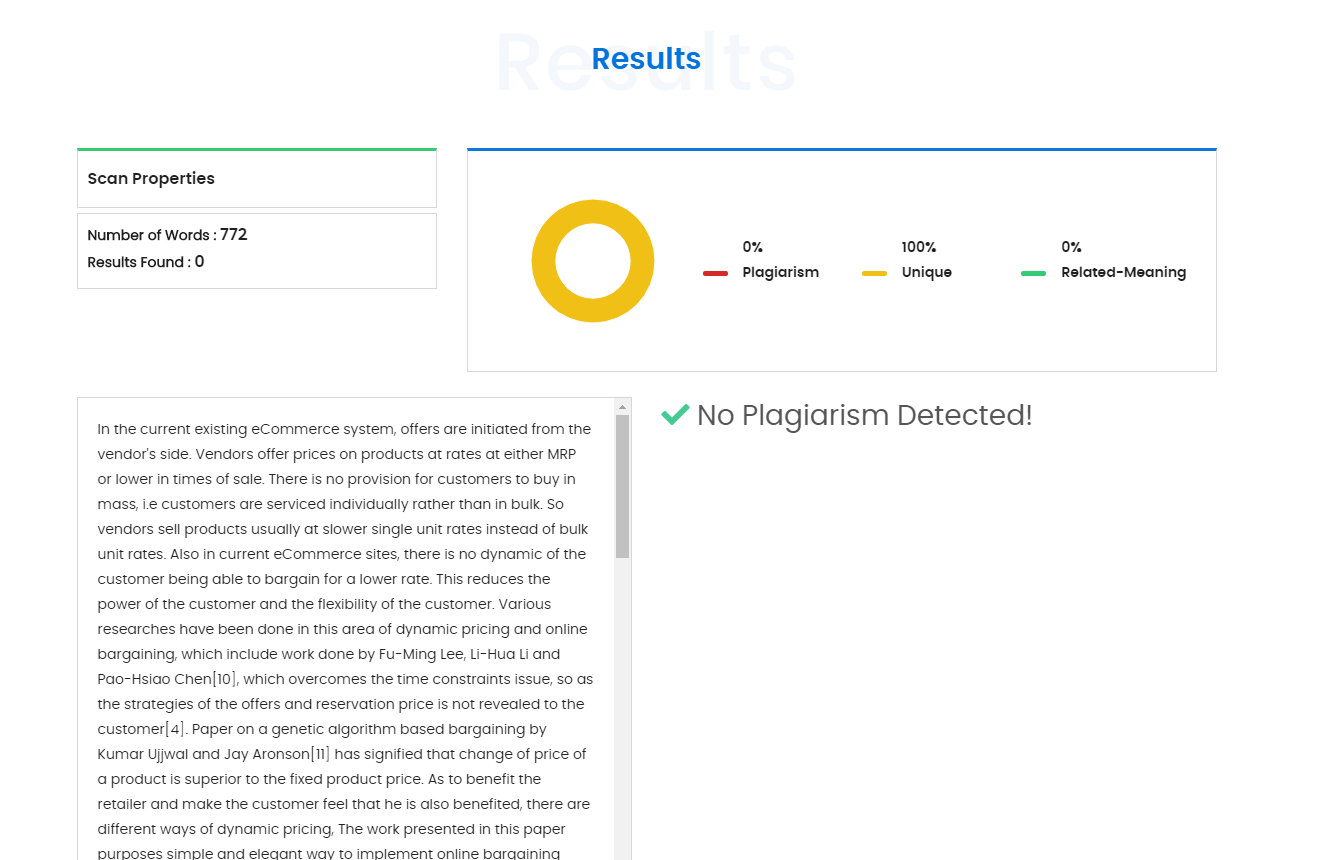
Frontend



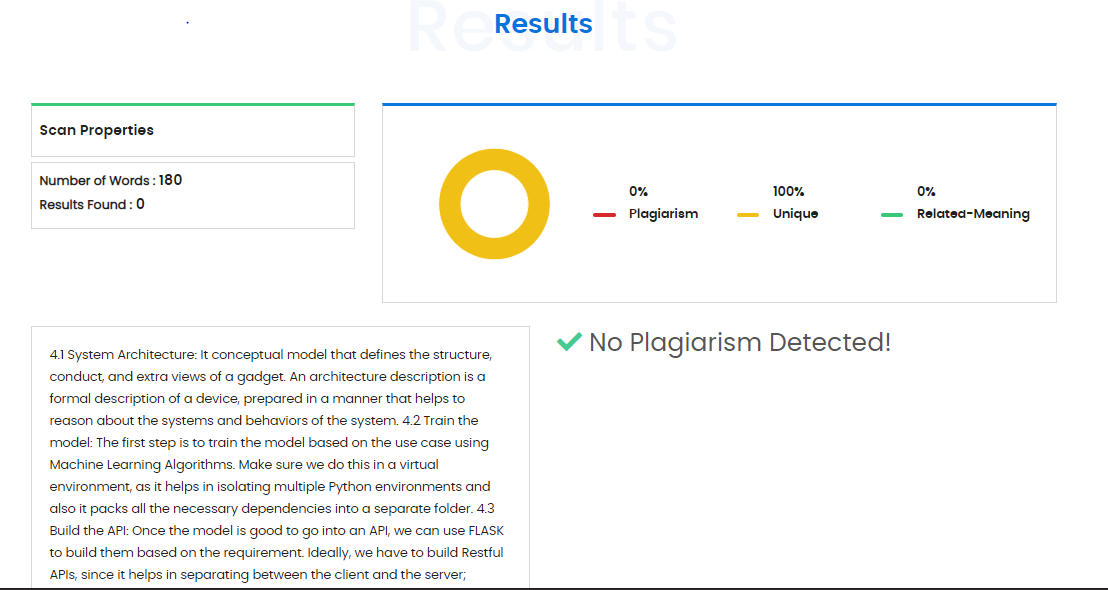
Backend

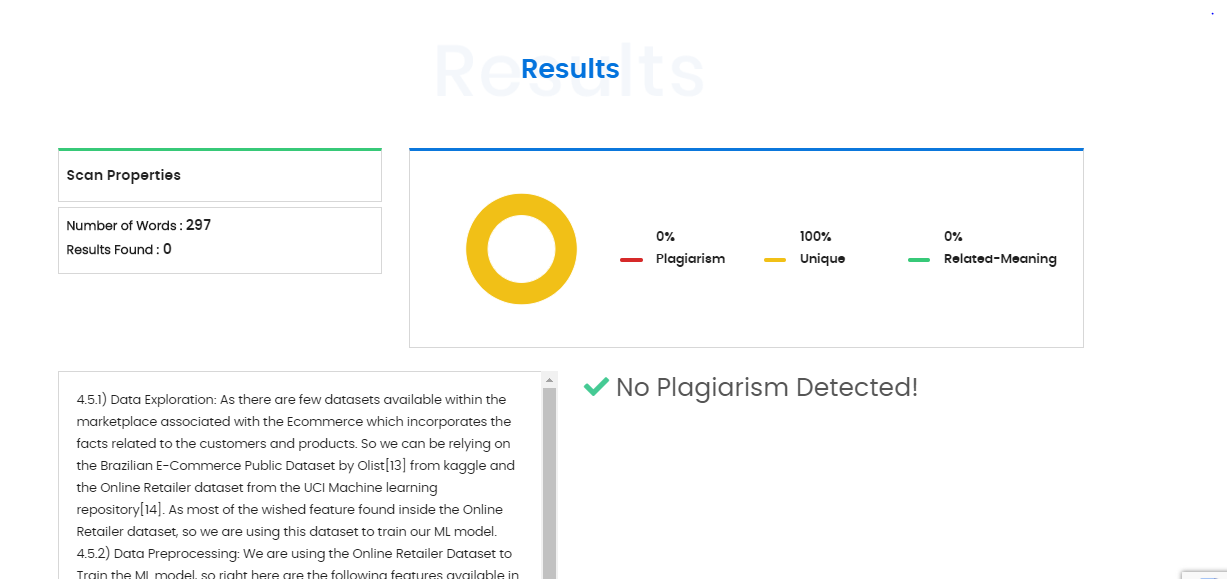


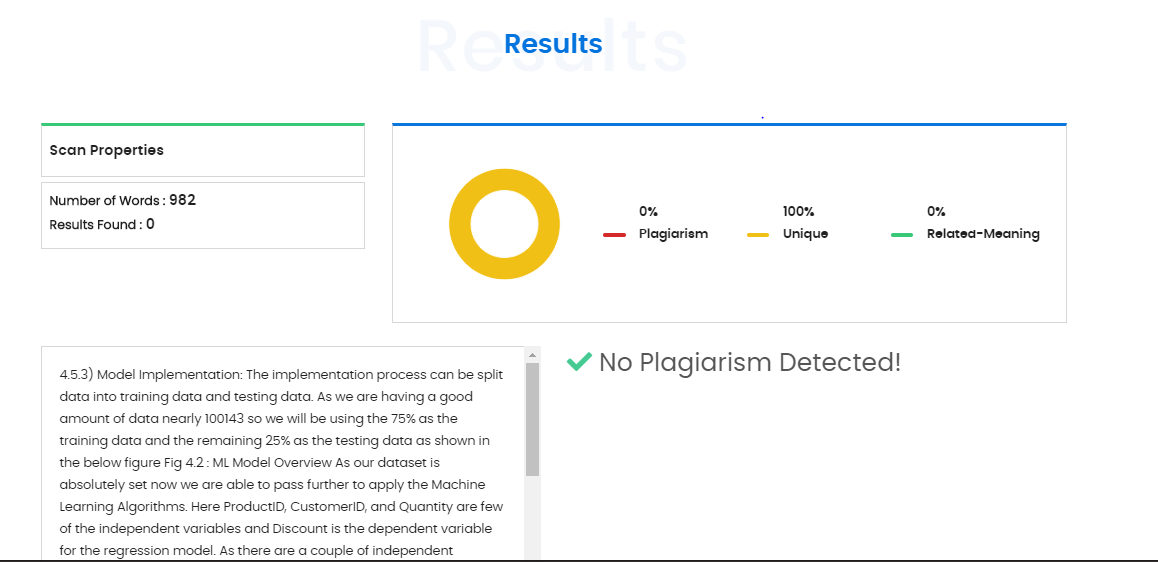
***Existing System***

****

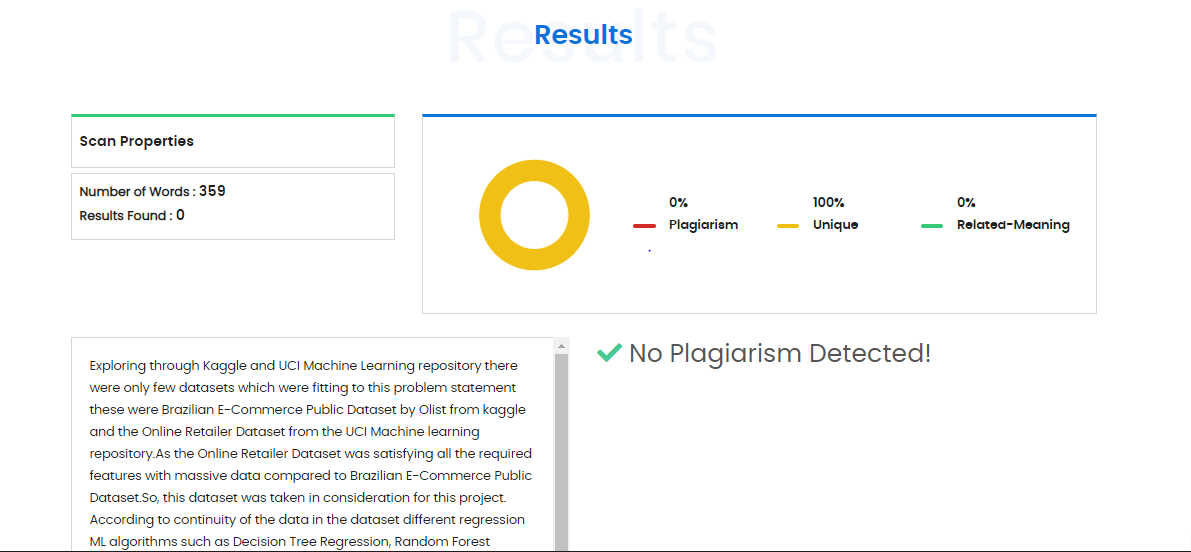
***Methodology***

****

****

****

***Summary***

****

**Student Details:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll No** | **Seat No** | **Email ID** | **Contact No** |
| Chayan Bobra | 09 | 807028 | chayanbobra13@gmail.com | 9479525465 |
| Kartik Jain | 10 | 807129 | 12kjain12@gmail.com | 9950745482 |
| Kunal Burgul | 11 | 807046 | kunal.burgul@gmail.com | 9420262640 |
| Sujay Khandare | 12 | 807110 | sujay.khandare@gmail.com | 9860771516 |
| Amol Dhole | 71 | 224013 | amoldhole1234@gmail.com | 8483963648 |

**Group Photo:**

****

**Github URL:**<https://github.com/kunalburgul/ChatBot-To-Bargain-Price-For-WooCommerce-Portal>