

INDUSTRY SKILLS RECCOMENDATION SYSTEM USING MACHINE LEARNING

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

ARMAAN RAUT (Reg.No.37110048)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND
TECHNOLOGY(DEEMED TO BE
UNIVERSITY)**

**Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by
AICTE**

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

March - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC | 12B Status by
UGC | Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600

119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING **BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **Mr. Armaan Raut (Reg.No.37110048)** who carried out the project entitled **"INDUSTRY SKILLS RECCOMENDATION SYSTEM USING MACHINE LEARNING"** under my supervision from November 2020 to March 2021.

Internal Guide

Mrs M.D.Anto Praveena M.C.A.,M.E.,(Ph.D)

Head of the Department

Dr.S.VIGNESHWARI M.E.,Ph.D.,

Dr.L.LAKSHMANAN M.E.,Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **Armaan Raut (Reg.No.37110048)** hereby declare that the Project Report entitled **“Industry Skills Recommendation System Using Machine Learning”** done by me under the guidance of **Mrs M.D.Anto Praveena M.C.A.,M.E.,(Ph.D).** Department of Computer Science and Engineering is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Technology.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D .**, Dean, School of Computing and **Dr.L.Lakshmanan M.E., Ph.D.,and Dr.S.Vigneshwari M.E., Ph.D.**, Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs M.D.Anto Praveena M.C.A.,M.E.,(Ph.D)** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Today's industry is very competitive, keeping up with the changing trends, technologies, and employment scenarios could become hectic at times. The saying in the industry today is keep learning new skills to keep yourself relevant in the current market. However, one of the biggest problem faced by people in this industry is not knowing what is worth of their skills. As everyone in today's industry knows the most important thing is to constantly learn and know what ones skills are worth in the current market. Hence, during job interviews people tend to get confused or flustered when faced with the question of what salary are you expecting. For people with experience there is a possibility that this question is not that difficult. However, in some cases people ask too much or too less and then regret their answer. Here is where this project comes to solve your problem with understanding where you and your skills stand in the current industry. The project's goal is to build a recommendation system that gives them a recommend salary they can expect from a company they are interviewing from based on the skills and job role they are applying for. It can help users to manage different industrial skills they have and to help them determine their worth in this competitive industry. This is done with the help of regression algorithms like random forest regression where users with the help of a web application select multiple skills that they know and get an estimate of the salary they can expect with the skills that they posses. Moreover, as mentioned earlier to help users with ease in access the project will be deployed in a web based application.

TABLE OF CONTENTS

S.NO.	TITLE	PAGE NO.
	ABSTRACT	i
	CONTENTS	ii
	LIST OF FIGURES	iv
	LIST OF ABBREVIATIONS	v
1.	INTRODUCTION	1
	DOMAIN INTRODUCTION	1
	OBJECTIVE	1
	PROBLEM DEFINITION	2
2.	LITERATURE SURVEY	3
	RELATED WORK	3
3.	AIM AND SCOPE OF PRESENT INVESTIGATION	4
	AIM OF THE PROJECT	4
	OBJECTIVE AND SCOPE	4
4.	SYSTEM IMPLEMENTATION	5
	SYSTEM ARCHITECHTURE	5
	METHODS AND MODEL DETAILS	6
	WEB SCRAPING	6
	MODEL DEVELOPMENT	8

WEB APPLICATION DEPLOYEMENT	13
5. RESULT AND DISCUSSION	15
EXPLORATORY DATA ANALYSIS	15
FINAL DEPLOYED APPLICATION	19
6. SUMMARY AND CONCLUSION	22
CONCLUSION	22
7. APPENDIX	24
A) SOURCE CODE	24
B) REFERENCES	36

LIST OF FIGURES

Fig. No.	Title	Page No.
1.	Fig 4.1: Architecture Diagram	5
2.	Fig 4.2.1: Basic Web Scraping Architecture	6
3.	Fig 4.2.2.a: Random Forest Regresor	10
4.	Fig 4.2.2.b: Decision Tree	12
5.	Fig 5.1.a: Salary difference between junior and senior job offers	15
6.	Fig 5.1.b: Job description differences	16
7.	Fig 5.1.c: Boxplots between age, average salary, and rating	16
8.	Fig 5.1.d: Factors comparison	17
9.	Fig 5.1.e: Graph of location	18
10.	Fig 5.1.f: Graph of industries	18
11.	Fig 5.1.g: Graph of Companies	19
12.	Fig 5.2.a: Running the application	20
13.	Fig 5.2.b: The application page	20
14.	Fig 5.2.c: The application page	21
15.	Fig 5.2.d: Recommended salary for the user	21

LIST OF ABBREVIATIONS

S. NO.	ACRONYM	EXPANSION
1.	ML	MACHINE LEARNING
2.	RFR	RANDOM FOREST REGRESSION
3.	MLR	MULTIPLE LINEAR REGRESSION
4.	LR	LASSO REGRESSION
5.	EDA	EXPLORATORY DATA ANALYSIS

CHAPTER 1

INTRODUCTION

DOMAIN INTRODUCTION

The current industry is not only full of job opportunities but also full of competition. The only way to keep on top of the competition is by keeping yourself up to date with the current changing trends and technologies. Professionals face a lot of confusion during interviews in this ever developing and changing industry standards. It is important for professionals to understand and remember the worth of their skills. During interviews many people face the problem with one particular question, that is “what salary are you expecting?”. Although professionals with a lot of experience mostly do not worry about this when they are interviewing for new jobs as they are looking for a pay better than what they have. However, for freshers, and professionals with reflectively less experience answering too much or too little to the answer can leave a bad impression. Hence, to help people estimating the correct worth of their skills and experience is what the main goal of this project. The goal is achieved with the help of regression machine leaning algorithms. The data is scraped from top job searching websites like Glassdoor. Lastly is it to be deployed for ease of access to the users in a web application format.

OBJECTIVE

Main focus of the project is to build a recommendation system that will take user input of the various skills they posses and help individuals to understand how much compensation they can expect by the details they provide. Target audience for this project is professionals who are unsure of what is their worth in the current market, what type of salary they can expect from an interview, and people who want to prepare for an upcoming job interview by analyzing how much salary they can ask for based o their skills without making a bad impression. To pull this off the project is using machine leaning technology with the help of regression algorithms

such as, multiple linear regression, lasso regression, and most importantly random forest. The model will be trained based on the data acquired by web scraping done on Glassdoor. At last they model will be deployed in an user friendly manner in the form of a web application.

PROBLEM DEFINITION

The Project's problem statement is, "To build a machine learning model which can take inputs based on different industry skills, company details, and job details. Moreover, it will provide the users with and output of what is the recommended offer the users can expect during an interview based on the details of their skills and company they provided." In simpler terms this project will deploy a web application that runs on a machine learning model that will help users who are applying for a job vacancy. It will help them to understand what is the worth of the skills they have acquired in the current industry. On top of that it will also help users to prepare for their upcoming interview for a particular position based on the compensation they can expect.

CHAPTER 2

LITERATURE SURVEY

2.1 RELATED WORK

Machine Learning is basically that field of computer science with the help of which computer systems can provide sense to data in much the same way as human beings do. The key focus of machine learning is to allow computer systems to learn from experience without being explicitly programmed or human intervention [11]. Here the proposed process is to broadly divided into few steps. It starts with first scraping the data from the website Glassdoor. Manually coping the data from website to local computer file storage is a very exhausting task and time consuming. Due to such limitation web scraping techniques are introduces. By using web scraping techniques user can extract information available on multiple website into a single database or spreadsheets. So data can be easily visualize and analyse for further use. Web scraping technique is a sub-discipline of web mining technology [2]. The collected data needs to be cleaned in order to be used, this is because the raw data acquired can be very hard to read and analyse. Once the data is collected and cleaned it can be used as the data set to build the model that is needed for the project. Now as the project use regression model for recommendation there will be multiple algorithms used to determine which give the highest accuracy. Once the model is successfully built and the desired output is acquired on the basis of the input provided by the users the model is needed to deployed in some form. For this project the plan is to deploy the model in the form of a web application. It is done with the help of a python based library known as streamlit. It is the easiest and quickest way to build web apps that can present your ML models and data attractively using it's awesome UI elements and Markdown, The deployment is simple, fast, and most importantly in Python. However, there are a couple of challenges with it. As it uses Google colab as the backend to build and as you might be aware, the colab session automatically restarts after 12 hours.

CHAPTER 2

AIM AND SCOPE OF PRESENT INVESTIGATION

AIM OF THE PROJECT

The main aim of this project is to design an effective recommendation system for providing an estimated salary based on the skills and details of the company and job description. To build a machine learning model which can take inputs based on different industry skills, company details, and job details. Moreover, it will provide the users with an output of what is the recommended offer the users can expect during an interview based on the details of their skills and company they provided. All this is done by taking data from Glassdoor and then using the extracted data for the model built with random forest regression. At last deploying it in form of a web application.

OBJECTIVE AND SCOPE

To recommend an estimated salary based on the user input of the skills they know, the company they are applying for, and the job description. It aims to analyse data on popular websites like Glassdoor and analyse them in order to give users an insight on what is the worth of their skills in the current market. Moreover, it provides the users that much needed knowledge before an interview to make sure they leave a great impression when faced with the question of their salary expectations.

CHAPTER 4

SYSTEM IMPLEMENTATION

SYSTEM ARCHITECTURE

The proposed system aims to analyse the data collected by the website Glassdoor and to recommend an estimated salary that users can expect when applying for a job vacancy. The steps for the working of the projects are as follows:

- Collection of data by web scraping
- Cleaning the raw data
- Supervised learning under random forest regression for recommendation
- Exploratory Data Analysis
- Deployment in form of a web application

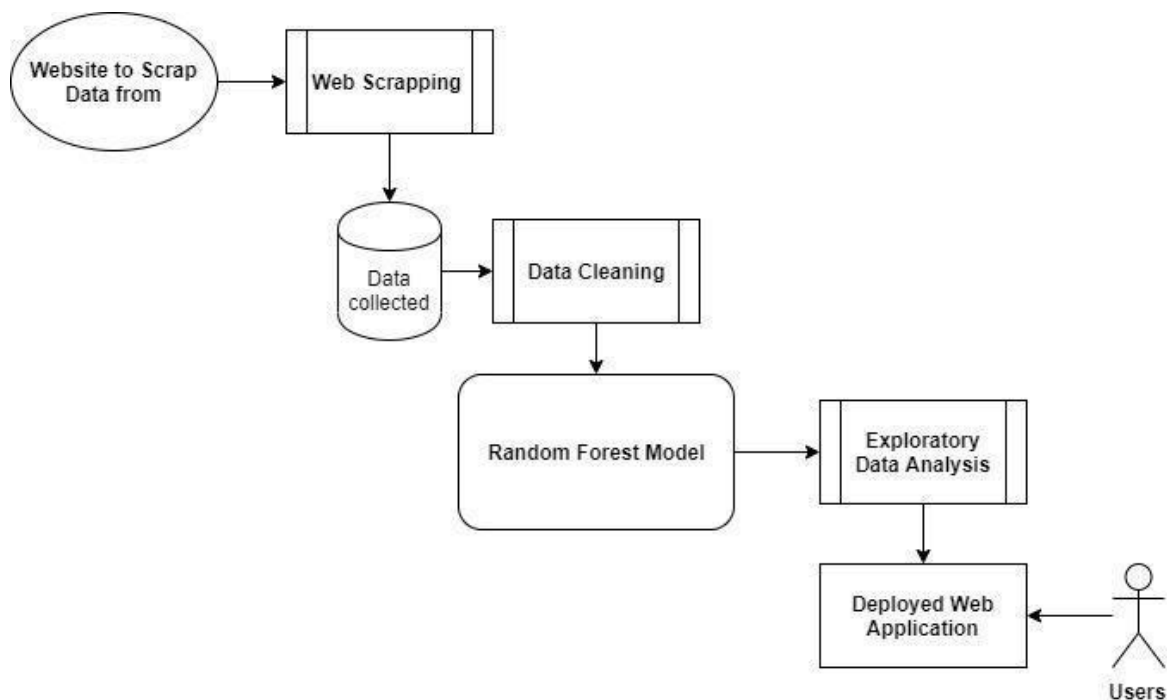


Figure 4.1: Architecture Diagram

METHODS AND MODEL DETAILS:

This project very briefly can be summarized into three categories:

- Web scraping
- Model development
- Web application deployment

Web Scraping

In the field of marketing, scientific or academic research data plays an important role. Researcher, market analyzer or academicians gather data from different websites for their better improvement. Copying of data on the website to user local storage is forbidden by most of the website authority. So that the user wants to manually copy the data from website to local computer file storage. But such a task is very exhausting and time consuming. Due to such limitation web scraping techniques are introduced. By using web scraping techniques user can extract information available on multiple websites into a single database or spreadsheets. So data can be easily visualized and analyzed for further use. Web scraping technique is a sub-discipline of web mining technology. The main objective of this paper is to review on the different web scraping techniques and software which can be used to extract required information from web sites. Web Mining stays at the crossroads of Information Retrieval, Information Extraction and Data Mining. Both Information Retrieval and Information Extraction have important roles to trace and mine valuable information out of unstructured data, before it is suitable to be processed by data mining applications. Exploring better these techniques is extremely necessary to cope with the amount of available data in the Information Overload Era. Also, with the Web being increasingly oriented towards the importance of semantics and integration of information, these areas of study become very important to address the new future trends of the Web.



Figure 4.2.1: Basic Web Scraping Architecture

Now in this project we will be using a python based web scraping library known as selenium to carry on the web scraping process. Selenium is an open-source web-based automation tool. Selenium primarily used for testing in the industry but It can also be used for web scraping. We'll use the Chrome browser but you can try on any browser, It's almost the same. For this project we will use Python3.x. You can also use *Python2.x* but there may be some slight differences in the code. Additionally we will also need the following packages:

- Selenium package
- ChromeDriver
- Virtualenv
- Extras

To get automatic information from website, web scraping is the most effective and efficient technique. Among all other techniques mention in this paper which are used to extract and store data, web scraping is more reliable, fast and automatic data retrieval system. By using web scraping terminology user can easily extract unstructured data on single or multiple websites into a structured data automatically. The main aim of this technique is to get information from web and aggregate into a new data set. Once we have collected the data set we needed we now need to clean this raw data into a readable data that can be analyzed using the machine learning model that we will be building. Data scientists spend a large amount of their time cleaning data sets and getting them down to a form with which they can work. In fact, a lot of data scientists argue that the initial steps of obtaining and cleaning data constitute 80% of the job. In this project we'll leverage Python's Pandas and NumPy libraries to clean data. Often, you'll find that not all the categories of data in a data set are useful to you. For example, you might have a data set containing student information (name, grade, standard, parents' names, and address) but want to focus on analyzing student grades. In this case, the address or parents' names categories are not important to you. Retaining these unneeded categories will take up unnecessary space and potentially also bog down run-time. Pandas provides a handy way of removing unwanted columns or rows from a Data Frame with the `drop()` function. A Pandas Index extends the functionality of NumPy arrays to allow for more versatile slicing and labeling. In

many cases, it is helpful to use a uniquely valued identifying field of the data as its index. Once our data set is cleaned and ready to be analyzed we will move to the next stage of the project which is also the most important stage of the project.

Model Development

Model development is the main part of this project here we will test out various regression algorithms in order to determine which particular regression model has the best accuracy and the best outcome

Multiple Linear Regression: It's a form of linear regression that is used when there are two or more predictors. It is the most common form of linear regression analysis. As a predictive analysis, the multiple linear regression is used to explain the relationship between one continuous dependent variable and two or more independent variables. The independent variables can be continuous or categorical

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$$

Here, Y is the output variable, and X terms are the corresponding input variables. Notice that this equation is just an extension of Simple Linear Regression, and each predictor has a corresponding slope coefficient (β).

The first β term (β_0) is the intercept constant and is the value of Y in absence of all predictors (i.e., when all X terms are 0). It may or may not hold any significance in a given regression problem. It's generally there to give a relevant nudge to the line/plane of regression.

Regression residuals must be normally distributed. A linear relationship is assumed between the dependent variable and the independent variables. The residuals are homoscedastic and approximately rectangular-shaped. Absence of multicollinearity is assumed in the model, meaning that the independent variables are not too highly correlated. At the centre of the multiple linear regression analysis is the task of fitting a single line through a scatter plot. More specifically the multiple linear regression fits a line through a multi-dimensional space of data points. The simplest form has one dependent and two independent variables. The dependent

variable may also be referred to as the outcome variable or regressand. The independent variables may also be referred to as the predictor variables or regressors.

There are 3 major uses for multiple linear regression analysis. First, it might be used to identify the strength of the effect that the independent variables have on a dependent variable. Second, it can be used to forecast effects or impacts of changes. That is, multiple linear regression analysis helps us to understand how much will the dependent variable change when we change the independent variables. Third, multiple linear regression analysis predicts trends and future values. The multiple linear regression analysis can be used to get point estimates. When selecting the model for the multiple linear regression analysis, another important consideration is the model fit. Adding independent variables to a multiple linear regression model will always increase the amount of explained variance in the dependent variable (typically expressed as R^2). Therefore, adding too many independent variables without any theoretical justification may result in an over-fit model.

After performing multiple linear regression we saw and come to a conclusion that it is not the best model as the accuracy received is of -23.941355441378295 and we can achieve better outcomes from other algorithms.

Lasso Regression: Lasso regression is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters). This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination. The acronym "LASSO" stands for Least Absolute Shrinkage and Selection Operator. Lasso regression performs L1 regularization, which adds a penalty equal to the absolute value of the magnitude of coefficients. This type of regularization can result in sparse models with few coefficients; Some coefficients can become zero and eliminated from the model. Larger penalties result in coefficient values closer to zero, which is the ideal for producing simpler models. On the other hand, L2 regularization (e.g. Ridge regression) *doesn't* result in elimination of coefficients or

sparse models. This makes the Lasso far easier to interpret than the Ridge. Lasso solutions are quadratic programming problems, which are best solved with software (like Matlab). The goal of the algorithm is to minimize:

$$\sum_{i=1}^n (y_i - \sum_j x_{ij} \beta_j)^2 + \lambda \sum_{j=1}^p |\beta_j|$$

After performing lasso regression we saw and come to a conclusion that it is not the best model as the accuracy received is of -23.92235957033688 and we can achieve better outcomes from other algorithms.

Random Forest Algorithm: Random forest algorithm is a flexible machine learning algorithm that produces great results even without hyper-parameter tuning. Apart from being simple to use, it is extremely accurate also. It is basically a supervised learning algorithm. A large number of decision trees operate together; an individual tree in a random forest model gives some prediction and finally the one with most votes becomes the prediction of the model. Random forest can be used for both classification as well as regression problems. A large number of relatively uncorrelated trees operating as a committee will perform better than individual constituent tree. Uncorrelated trees can produce better and more accurate results than individual decision trees, the reason for this is that the trees protect each other from their individual errors, thus even if some trees in the group are wrong, all the trees are able to move in correct direction given that many other trees will be right.

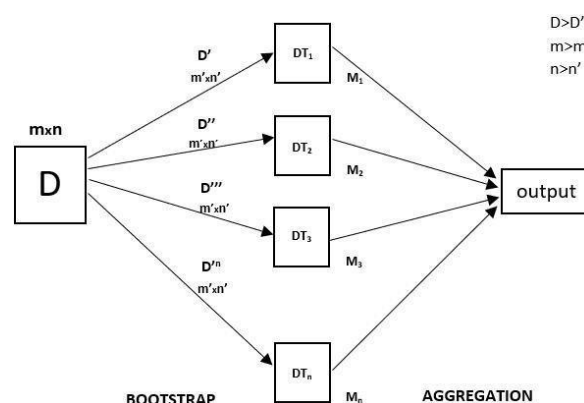


Figure 4.2.2.a Random Forest Regresor

Random Forest works in four steps:

- Select random samples from a given dataset.
- Construct a decision tree for each sample and get a prediction result from each decision tree.
- Perform a vote for each predicted result.
- Select the prediction result with the most votes as the final prediction.

Decision Trees:

The classification technique is a systematic approach to build classification models from an input dataset. For example, decision tree classifiers, rule-based classifiers, neural networks, support vector machines, and naive Bayes classifiers are different technique to solve a classification problem. Each technique adopts a learning algorithm to identify a model that best fits the relationship between the attribute set and class label of the input data. Therefore, a key objective of the learning algorithm is to build predictive model that accurately predict the class labels of previously unknown records.

Decision Tree Classifier is a simple and widely used classification technique. It applies a straightforward idea to solve the classification problem. Decision Tree Classifier poses a series of carefully crafted questions about the attributes of the test record. Each time it receives an answer, a follow-up question is asked until a conclusion about the class label of the record is reached.

Build an optimal decision tree is key problem in decision tree classifier. In general, may decision trees can be constructed from a given set of attributes. While some of the trees are more accurate than others, finding the optimal tree is computationally infeasible because of the exponential size of the search space.

However, various efficient algorithms have been developed to construct a reasonably accurate, albeit suboptimal, decision tree in a reasonable amount of time. These algorithms usually employ a greedy strategy that grows a decision tree by making a series of locally 17 optimum decisions about which attribute to use for partitioning the data. For example, Hunt's algorithm, ID3, C4.5, CART, SPRINT are greedy decision tree induction algorithms.

The decision tree inducing algorithm must provide a method for specifying the test condition for different attribute types as well as an objective measure for evaluating the goodness of each test condition.

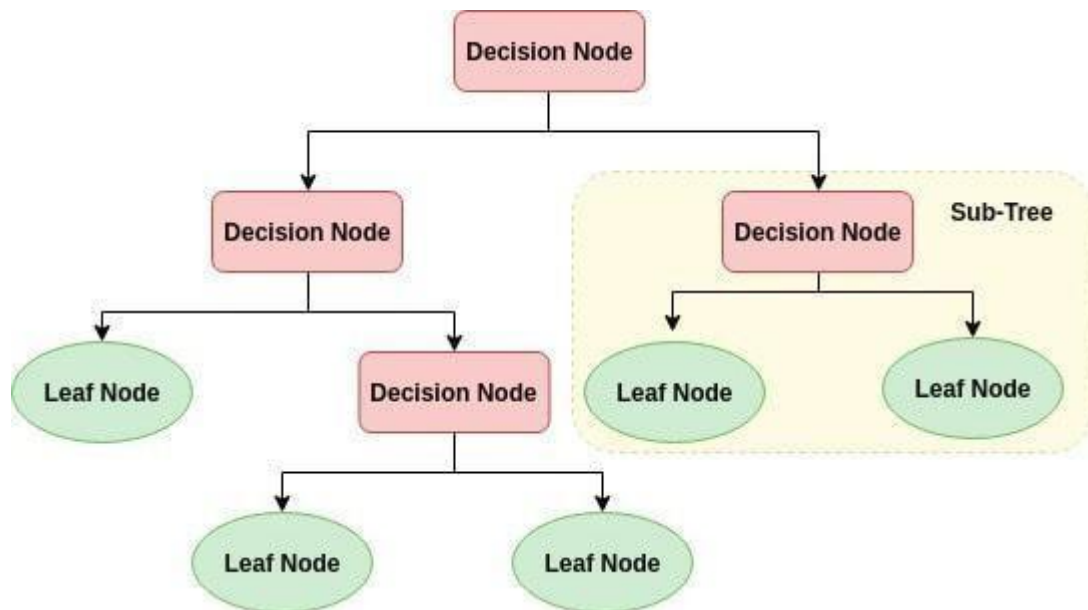


Figure 4.2.2.b: Decision Tree

First, the specification of an attribute test condition and its corresponding outcomes depends on the attribute types. We can do two-way split or multi-way split, discretize or group attribute values as needed. The binary attributes lead to two-way split test condition. For nominal attributes which have many values, the test condition can be expressed into multi way split on each distinct value, or two-way split by grouping the attribute values into two subsets. Similarly, the ordinal attributes can also produce binary or multi way splits as long as the grouping does not violate the order property of the attribute values. For continuous attributes, the test condition can be expressed as a comparison test with two outcomes, or a range query. Or we can discretize the continuous value into nominal attribute and then perform two-way or multi-way split.

Since there are many choices to specify the test conditions from the given training set, we need use a measurement to determine the best way to split the records. The goal of best test conditions is whether it leads a homogenous class distribution in the nodes, which is the purity of the child nodes before and after splitting. The larger the degree of purity, the better is the class distribution.

To determine how well a test condition performs, we need to compare the degree of impurity of the parent before splitting with degree of the impurity of the child nodes after splitting. The larger their difference, the better is the test condition. The measurements of node impurity/purity are:

- Gini Index
- Entropy
- Misclassification Error

$$\text{Gini index: } G(E) = 1 - \sum_{j=1}^C p_j^2,$$

$$\text{entropy: } H(E) = - \sum_{j=1}^C p_j \log p_j.$$

With random forest regression we are able to get an accuracy of -23.066594032252755. Which is the best result so far hence we are concluding that we got the best possible accuracy with random forest regression and we are moving forward with the random forest regression model.

Web Application Deployment

As per the founders of Streamlit, it is the fastest way to build data apps and share them. It is a recent model deployment tool that simplifies the entire model deployment cycle and lets you deploy your models quickly. It is a simple, quick, and interpretative model deployment tool.

Here are some of the key features of Streamlit:

1. It quickly turns data scripts into shareable web applications. You just have to pass a running script to the tool and it can convert that to a web app.
2. Everything in Python. The best thing about Streamlit is that everything we do is in Python. Starting from loading the model to creating the frontend, all can be done using Python.
3. All for free. It is open source and hence no cost is involved. You can deploy your apps without paying for them.

No front-end experience required. Model deployment generally contains two parts, frontend, and backend. The backend is generally a working model, a machine learning model in our case, which is built-in python. And the front end part, which generally requires some knowledge of other languages like java scripts, etc. Using Streamlit, we can create this front end in Python itself. So, we need not learn any other programming languages or web Let's look at the deployment pipeline if we use Streamlit:

- Model Building
- Creating a python script
- Create front-end: Python
- Deploy

Here we will build the model and create a python script for it. Then we will build the front-end for the app which will be in python and finally, we will deploy the model.

CHAPTER 5

RESULT AND DISCUSSION

EXPLORATORY DATA ANALYSIS

The final result is a fully developed web application that can show a recommended salary when the user provide multiple inputs like skills, company, and job details. Before the final look into the web application we will observe the exploratory data analysis of the project. Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

To begin with we will observe a few of the graphical outputs from the project.

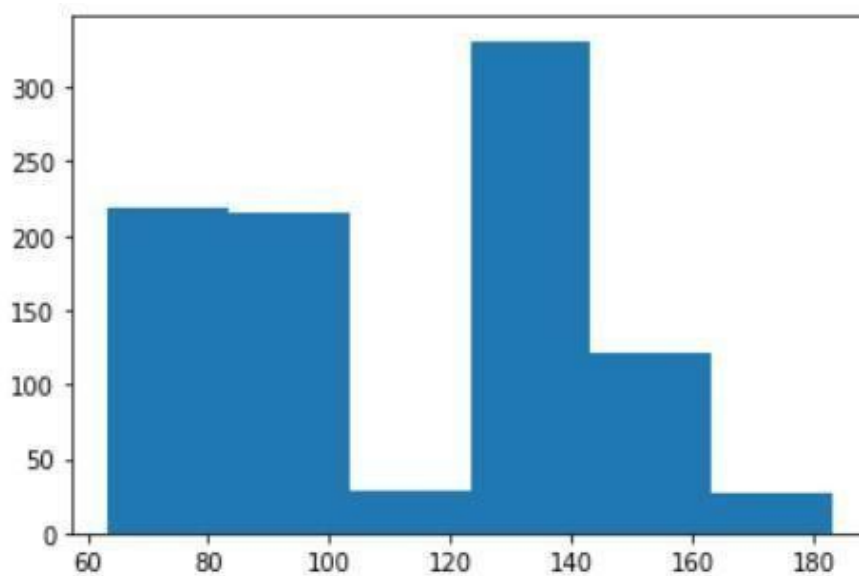


Figure 5.1.a: Salary difference between junior and senior job offers

This plot shows a lot of jobs in the \$60,000 to \$100,000 range and an even higher number in the \$120,000 to \$140,000 range, but very few jobs in the \$100,000 to \$120,000 range, which might indicate a clear distinction in salary between the junior and senior levels.

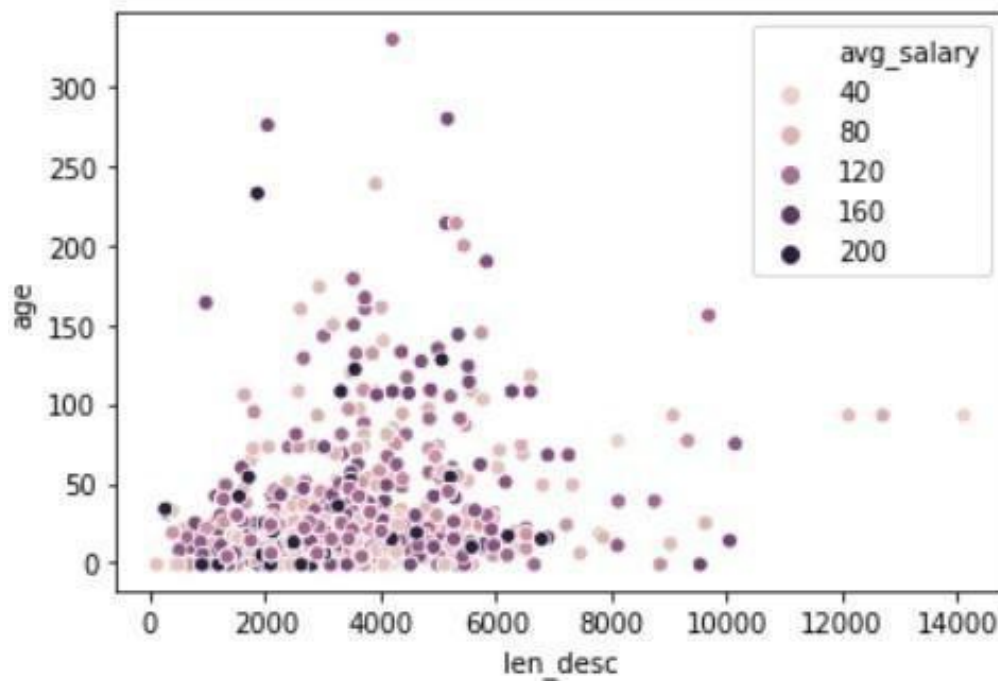


Figure 5.1.b: Job description differences

A very cluttered plot, the only thing to focus on are the outliers: The really long descriptions are jobs that probably won't pay as much as the others. The really old companies (200+ years) actually keep their descriptions short, and also pay on a higher scale typically.

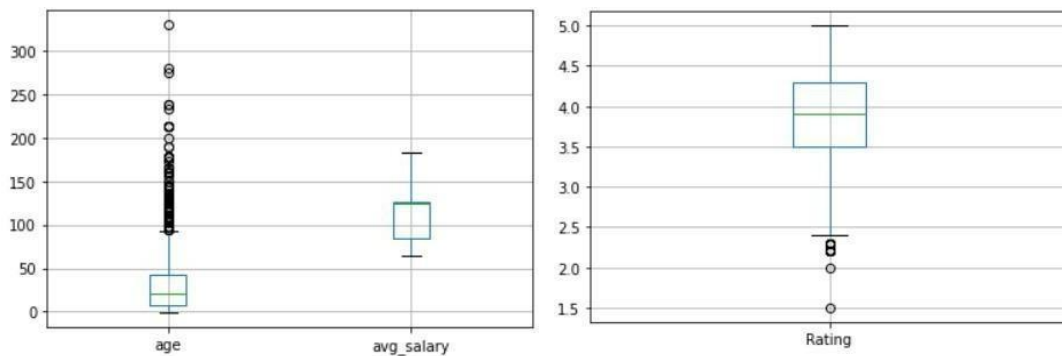


Figure 5.1.c: Boxplots between age, average salary, and rating

These boxplots show that age has a lot of outliers on the higher side, rating has a few on the lower side, while average salary has a very uniform distribution.

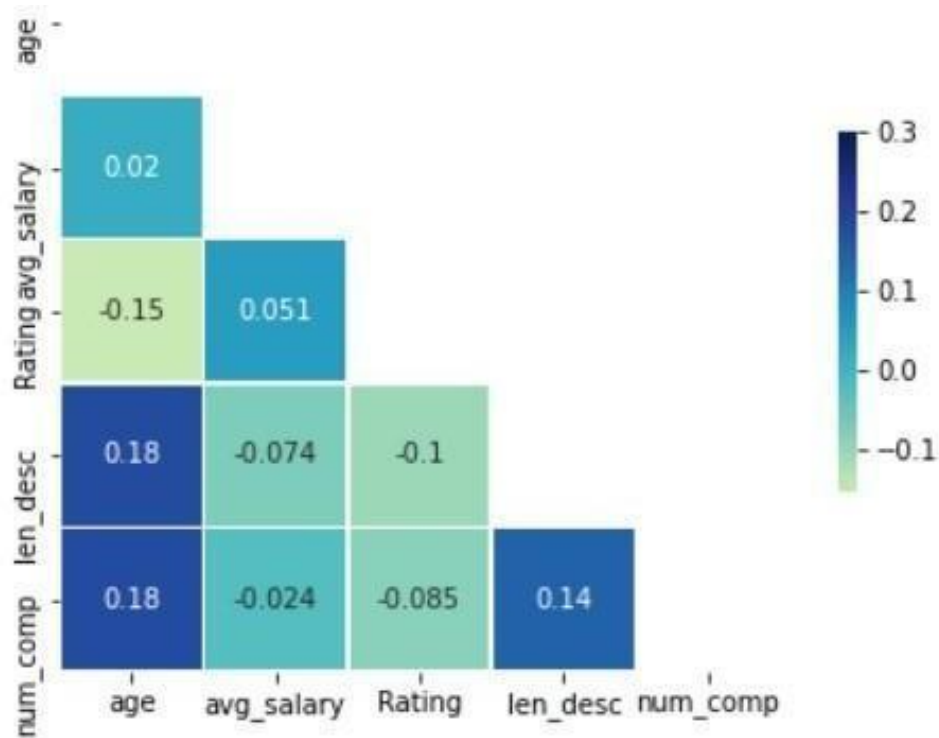


Figure 5.1.d: Factors comparison

Although there are no strong correlations, Description Length, Age and Number of Competitors seem to be slightly correlated.

The project had a lot of graphs that were complicated due to the vast amount of data collected and modulated. Here are some of the important graphs from the EDA

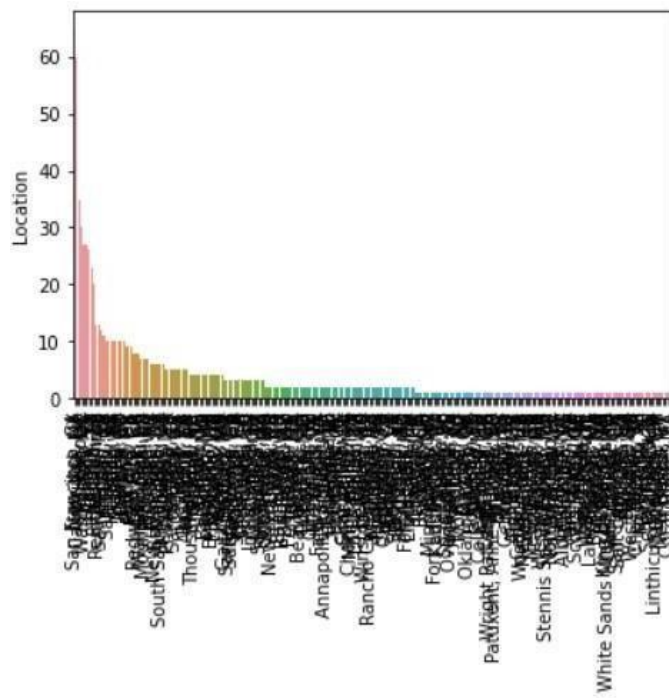


Figure 5.1.e Graph of location

There are a total of 243 locations taken into account and around 266 headquarters as well.

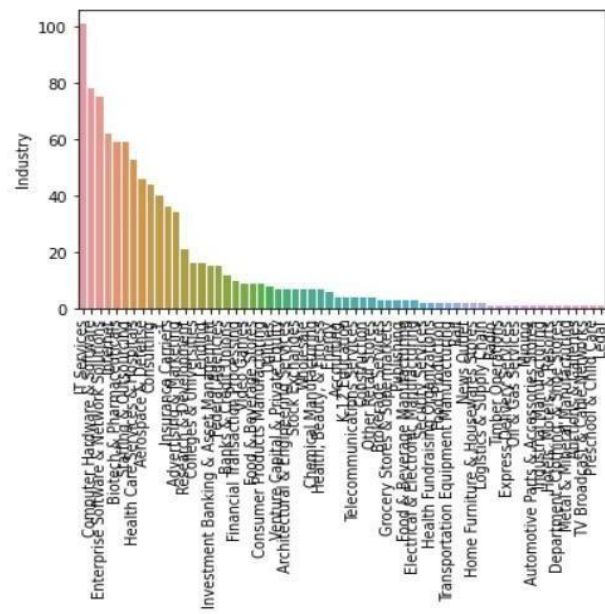


Figure 5.1.f: Graph of industries

There are around 62 industries 23 sectors taken as data from Glassdoor.

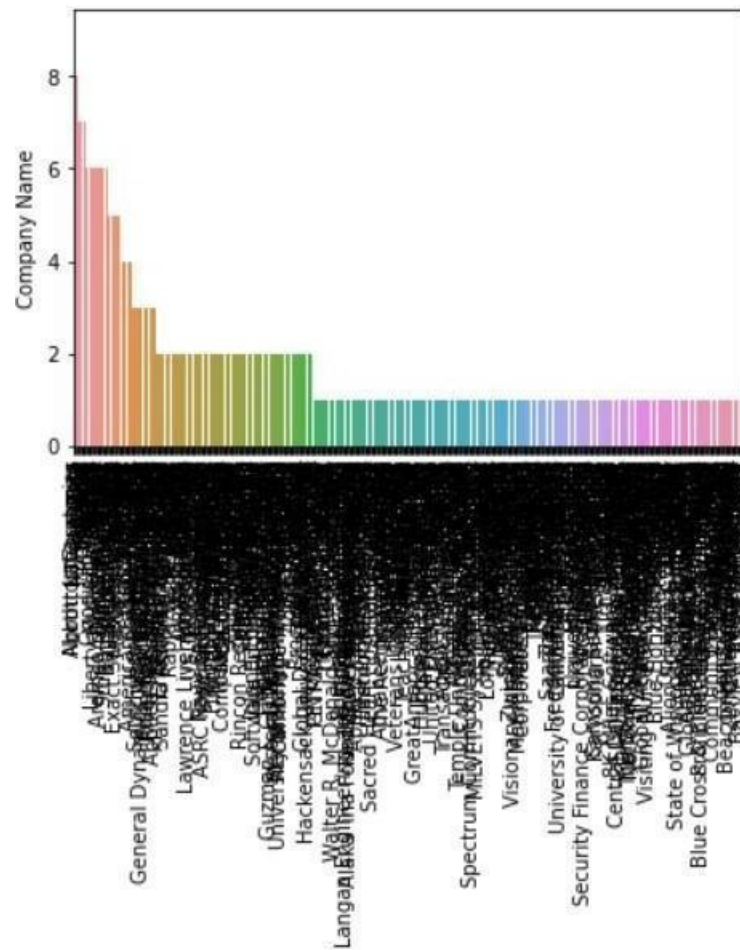


Figure 5.1.g: Graph of Companies

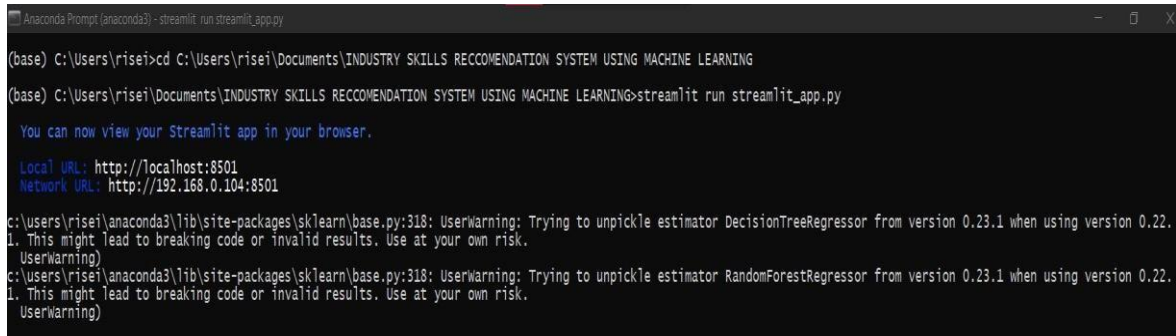
The above graph is of 547 different companies whose data is extracted or scraped from Glassdoor. This data as seen in previous graphs includes age, average salary, rating, job description, etc.

FINAL DEPLOYED APPLICATION

As we observed few of the many different EDA methods used on this project in order to understand the project in a more graphical manner now it is time to observe the final output. As mentioned earlier this project is deployed with the help of streamlit library. Moreover, it is the fastest way to build data apps and share them. It is a recent model deployment tool that simplifies the entire model deployment cycle and lets you deploy your models quickly. It is a simple, quick, and interpretative model deployment tool.

To run the application the process is quite similar to running any python program on your computer.

1. Change directory to the location of the program on your computer
2. Use the line `streamlit run streamlit_app.py`



```
Anaconda Prompt (anaconda3) - streamlit: run streamlit_app.py
(base) C:\Users\risei>cd C:\Users\risei\Documents\INDUSTRY SKILLS RECCOMENDATION SYSTEM USING MACHINE LEARNING
(base) C:\Users\risei\Documents\INDUSTRY SKILLS RECCOMENDATION SYSTEM USING MACHINE LEARNING>streamlit run streamlit_app.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.0.104:8501

c:\Users\risei\anaconda3\lib\site-packages\sklearn\base.py:318: UserWarning: Trying to unpickle estimator DecisionTreeRegressor from version 0.23.1 when using version 0.22.1. This might lead to breaking code or invalid results. Use at your own risk.
  UserWarning)
c:\Users\risei\anaconda3\lib\site-packages\sklearn\base.py:318: UserWarning: Trying to unpickle estimator RandomForestRegressor from version 0.23.1 when using version 0.22.1. This might lead to breaking code or invalid results. Use at your own risk.
  UserWarning)
```

Figure 5.2.a: Running the application

Once the anaconda prompt successfully runs the program you will see a web page appear in your default browser and the program will be up and running:

Industry Skills Recommendation System

All fields are mandatory.

Your skills:

- ☐ Python
- ☐ C Programming
- ☐ Java
- ☐ C++
- ☐ Strong Algorithmic Knowledge
- ☐ HTML CSS JS
- ☐ R
- ☐ AWS
- ☐ Spark
- ☐ Hadoop
- ☐ Docker
- ☐ SQL
- ☐ Linux
- ☐ Flask

Figure 5.2.b: The application page

Check Glassdoor for exact values, if unsure

Age of the Company

0.00 - +

Number of Competitors

0.00 - +

Details about the Job:

Is the Job at Headquarters? (0 for No, 1 for Yes)

☒ 0

☐ 1

Job Type

SW ▼

Senior role?

☒ Senior

☐ Not Senior

Character Length of the Job Description

0.00 - +

Reccomend

Figure 5.2.c: The application page

As one can observe there are a lot of inputs users can select to ideally get an estimated recommendation of the salary they can expect from a job vacancy. They have the option to select from a large list of skills, details about the company, and job.

Finally once the user selects all the inputs ideal to their situation they can click o the recommend button to get the result.

Reccomend

Your reccomended salary is Rs: 723750.0

Figure 5.2.d: Recommended salary for the user

CHAPTER 6

SUMMARY AND CONCLUSION

6.1 CONCLUSION

At the start of the project the goal was to solve this particular problem statement “To build a machine learning model which can take inputs based on different industry skills, company details, and job details. Moreover, it will provide the users with and output of what is the recommended offer the users can expect during an interview based on the details of their skills and company they provided.” As the project progressed it was observed that the project needed to be divided in to three broad sections. To tackle those sections namely web scraping, model building, and web application development the language and the approach chosen was python libraries. For web scraping to pull it off successfully a python library by the name of selenium was used. Once this data was obtained the next step in this section that took place was cleaning of the said data. This was done as the obtained data was raw and pretty hard to analyse and contained a lot of information that was not necessarily important for the project. So the data was cleaned with the help of python libraries like pandas and numpy. After the raw data was converted and stored and easy to analyze clean data we moved to the step of developing the regression model that can help us with the problem statement. This said model will use the data collected from web scraping for the analysis. First we tried to build the model using multiple linear regression which gave us a result but the accuracy was not satisfactory enough. In order to improve the accuracy and make the model for efficient by removing unnecessary features from the model we used lasso regression. As lasso regression gave us a relatively better accuracy but still a better accuracy can be achieved so we moved to random forest regression. As random forest regression needs to work with decision trees and is a flexible machine learning algorithm that produces great results even without hyper-parameter tuning. Apart from being simple to use, it is extremely accurate also. It is basically a supervised learning algorithm. A large number of decision trees operate together; an individual tree in a random forest model gives some prediction and finally the one with most votes becomes the prediction of the model. Random forest can be used for both classification as well as regression problems. So we got our desired

accuracy and found out The Random Forest model seems to have the least error, so that's the model that will be chosen.

Now that we have made the model and selected the ideal algorithm according to our project. Here is the accuracy we achieved with all the algorithms:

- Multiple linear regression: -23.941355441378295
- Lasso regression: -23.92235957033688
- Random forest regression: -23.066594032252755

Finally, the project was deployed in a web application form using the python library streamlit where it can be accessed and used by the users with ease. It takes input such as skills, company, and job details to give the ideal recommended salary that one should expect from a job offer according to their skills.

APPENDIX

A) SOURCE CODE:

DATA COLLECTION

```
import glassdoor_scraper  
as gsimport pandas as pd
```

```
path = "/home/vishnub/chromedriver"
```

```
df = gs.get_jobs('Data Scientist', 1000, False, path, 15)
```

```
df.head()
```

```
df.to_csv('glassdoor_jobs.csv', index=False)
```

GLASSDOOR SCRAPER

```
from selenium.common.exceptions import NoSuchElementException,  
ElementClickInterceptedException, StaleElementReferenceException
```

```
from selenium.webdriver.common.keys import Keys
```

```
from selenium import webdriver
```

```
import time
```

```
import pandas as pd
```

```
def get_jobs(keyword, num_jobs, verbose, path, slp_time):
```

```
    """Gathers jobs as a dataframe, scraped from Glassdoor"""
```

```
    # Initializing the webdriver
```

```
    options = webdriver.ChromeOptions()
```

```
    # Uncomment the line below if you'd like to scrape without a new Chrome  
    window every time.
```

```
    # options.add_argument('headless')
```

```

# Change the path to where chromedriver is in your home folder.
driver = webdriver.Chrome(executable_path=path, options=options)
driver.set_window_size(1120, 1000)

url = "https://www.glassdoor.com/Job/data-scientist-jobs-SRCH_KE0,14.htm"
# url =
"https://www.glassdoor.com/Job/jobs.htm?suggestCount=0&suggestChosen=false
&clickSource=searchBtn&typedKeyword=" + \

# keyword+"&sc.keyword="+keyword+"&locT=&locId=&jobType="

#url = 'https://www.glassdoor.com/Job/jobs.htm?sc.keyword="' + keyword +
"&locT=C&locId=1147401&locKeyword=San%20Francisco,%20CA&jobType=all&
fromAge=-1&minSalary=0&includeNoSalaryJobs=true&radius=100&cityId=-1&min
Rating=0.0&industryId=-1&sgocId=-1&seniorityType=all&companyId=-1&employer
Sizes=0&applicationType=0&remoteWorkType=0'

driver.get(url)
jobs = []

# If true, should be still looking for new jobs.
while len(jobs) < num_jobs:

    # Let the page load. Change this number based on your internet speed.
    # Or, wait until the webpage is loaded, instead of hardcoding it.
    time.sleep(slp_time)

    # Test for the "Sign Up" prompt and get rid of it.
    try:
        driver.find_element_by_class_name("selected").click()
    except ElementClickInterceptedException:
        pass

    time.sleep(.1)

```

```

try:
    # clicking to the X.
    driver.find_element_by_css_selector('[alt="Close"]').click()
    print(' x out worked')
except NoSuchElementException:
    print(' x out failed')
    pass

# Going through each job in this page
# jl for Job Listing. These are the buttons we're going to click.
job_buttons = driver.find_elements_by_class_name("jl")
for job_button in job_buttons:

    print("Progress: {}".format(
        "" + str(len(jobs)) + "/" + str(num_jobs)))
    if len(jobs) >= num_jobs:
        break

    try:
        job_button.click()
    except ElementClickInterceptedException:
        driver.execute_script(
            "window.scrollTo(0, document.body.scrollHeight);")
        print("Click")

    time.sleep(1)
    collected_successfully = False

while not collected_successfully:
    try:

```

```

        company_name = driver.find_element_by_xpath(
            '://div[@class="employerName"]').text
        location = driver.find_element_by_xpath(
            '://div[@class="location"]').text
        job_title = driver.find_element_by_xpath(
            '://div[contains(@class, "title")]').text
        job_description = driver.find_element_by_xpath(
            '://div[@class="jobDescriptionContent desc"]').text
        collected_successfully = True
    except:
        time.sleep(10)

    try:
        salary_estimate = driver.find_element_by_xpath(
            '://span[@class="gray salary"]').text
    except NoSuchElementException:
        salary_estimate = -1 # You need to set a "not found value. It's
important."

    try:
        rating = driver.find_element_by_xpath(
            '://span[@class="rating"]').text
    except NoSuchElementException:
        rating = -1 # You need to set a "not found value. It's important."

# Printing for debugging
if verbose:
    print("Job Title: {}".format(job_title))
    print("Salary Estimate: {}".format(salary_estimate))
    print("Job Description: {}".format(job_description[:500]))
    print("Rating: {}".format(rating))

```

```

        print("Company Name: {}".format(company_name))
        print("Location: {}".format(location))

    # Going to the Company tab...
    # clicking on this:
    # <div class="tab"
data-tab-type="overview"><span>Company</span></div>
    try:
        driver.find_element_by_xpath(
            '//*[@class="tab" and @data-tab-type="overview"]').click()

    max_attempts = 10

    while True:
        headquarters = driver.find_element_by_xpath(
            '//*[@class="infoEntity"]//label[text()="Headquarters"]//following-sibling::*').text

        if headquarters is not None:
            break

        else:
            time.sleep(0.5)
            max_attempts -= 1
            if max_attempts == 0:
                print("Cannot find headquarters.")
                headquarters = -1
                break

    # try:
    # <div class="infoEntity">

```

```

        #    <label>Headquarters</label>
        #    <span class="value">San Francisco, CA</span>
        # </div>

        #    headquarters = driver.find_element_by_xpath(
        #
        #'.//div[@class="infoEntity"]//label[text()='Headquarters']//following-sibling::*').text
        # except StaleElementReferenceException:
        #    break

    try:
        size = driver.find_element_by_xpath(

        #'.//div[@class="infoEntity"]//label[text()='Size']//following-sibling::*').text
        except NoSuchElementException:
            size = -1

    try:
        founded = driver.find_element_by_xpath(

        #'.//div[@class="infoEntity"]//label[text()='Founded']//following-sibling::*').text
        except NoSuchElementException:
            founded = -1

    try:
        type_of_ownership = driver.find_element_by_xpath(

        #'.//div[@class="infoEntity"]//label[text()='Type']//following-sibling::*').text
        except NoSuchElementException:
            type_of_ownership = -1

    try:
        industry = driver.find_element_by_xpath(

```

```
'./div[@class="infoEntity"]//label[text()="Industry"]//following-sibling::*).text
```

```
except NoSuchElementException:
```

```
    industry = -1
```

```
try:
```

```
    sector = driver.find_element_by_xpath(
```

```
'./div[@class="infoEntity"]//label[text()="Sector"]//following-sibling::*).text
```

```
except NoSuchElementException:
```

```
    sector = -1
```

```
try:
```

```
    revenue = driver.find_element_by_xpath(
```

```
'./div[@class="infoEntity"]//label[text()="Revenue"]//following-sibling::*).text
```

```
except NoSuchElementException:
```

```
    revenue = -1
```

```
try:
```

```
    competitors = driver.find_element_by_xpath(
```

```
'./div[@class="infoEntity"]//label[text()="Competitors"]//following-sibling::*).text
```

```
except NoSuchElementException:
```

```
    competitors = -1
```

```
# Rarely, some job postings do not have the "Company" tab.
```

```
except NoSuchElementException:
```

```
    headquarters = -1
```

```
    size = -1
```

```
    founded = -1
```

```
    type_of_ownership = -1
```

```
industry = -1
sector = -1
revenue = -1
competitors = -1
```

```
if verbose:
```

```
    print("Headquarters: {}".format(headquarters))
    print("Size: {}".format(size))
    print("Founded: {}".format(founded))
    print("Type of Ownership: {}".format(type_of_ownership))
    print("Industry: {}".format(industry))
    print("Sector: {}".format(sector))
    print("Revenue: {}".format(revenue))
    print("Competitors: {}".format(competitors))
```

```
print("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@")
```

```
jobs.append({"Job Title": job_title,
            "Salary Estimate": salary_estimate,
            "Job Description": job_description,
            "Rating": rating,
            "Company Name": company_name,
            "Location": location,
            "Headquarters": headquarters,
            "Size": size,
            "Founded": founded,
            "Type of ownership": type_of_ownership,
            "Industry": industry,
            "Sector": sector,
            "Revenue": revenue,
```

```

        "Competitors": competitors})
    # add job to jobs

    # Clicking on the "next page" button
    try:
        driver.find_element_by_xpath('.//li[@class="next"]//a').click()
    except NoSuchElementException:
        print("Scraping terminated before reaching target number of jobs.
        Needed {}, got {}".format(
            num_jobs, len(jobs)))
        break

    # This line converts the dictionary object into a pandas DataFrame.
    return pd.DataFrame(jobs)

```

STREAMLIT APP

```

import streamlit
as st import joblib

import numpy as np
import pandas as pd
import sklearn

df = pd.read_csv('glassdoor_jobs_cleaned.csv')

st.title('Industry Skills Recommendation System')

model = joblib.load('model.pkl')

st.markdown(
    "## All fields are mandatory.")

```

```

st.subheader('Your skills:')
python_yn = st.checkbox("Python ")
st.checkbox("C Programming ")
st.checkbox("Java ")
st.checkbox("C++ ")
algo_yn = st.checkbox("Strong Algorithmic Knowledge")
st.checkbox("HTML CSS JS ")
r_yn = st.checkbox("R ")
aws_yn = st.checkbox("AWS")
spark_yn = st.checkbox("Spark ")
hadoop_yn = st.checkbox("Hadoop")
docker_yn = st.checkbox("Docker ")
sql_yn = st.checkbox("SQL")
linux_yn = st.checkbox("Linux")
flask_yn = st.checkbox("Flask")
django_yn = st.checkbox("Django")
tensorflow_yn = st.checkbox("Tensorflow")
keras_yn = st.checkbox("Keras")
pytorch_yn = st.checkbox("PyTorch")
tableau_yn = st.checkbox("Tableau")
stats_yn = st.checkbox(
    "Strong Statistical Knowledge")

st.subheader('Company Details: \n Check Glassdoor for exact values, if unsure')

rating = 0.00
age = st.number_input('Age of the Company', step=1.0, min_value=0.0)
num_comp = st.number_input('Number of Competitors', step=1.0, min_value=0.0)

```

```
st.subheader('Details about the Job:')
```

```
jobhq = st.radio(
```

```
    "Is the Job at Headquarters? (0 for No, 1 for Yes)", options=[0, 1])
```

```
job_type_num = st.selectbox("Job Type",
```

```
                                options=['sw', 'director','analyst','data scientist', 'na',  
    'data engineer', 'research', 'mle', 'manager'])
```

```
#print(list(df["job_type"].unique()))
```

```
def number_simplifier(role):
```

```
    if role == "data scientist":
```

```
        return 3
```

```
    elif role == "data engineer":
```

```
        return 2
```

```
    elif role == "analyst":
```

```
        return 1
```

```
    elif role == "director":
```

```
        return 4
```

```
    elif role == "manager":
```

```
        return 5
```

```
    elif role == "mle":
```

```
        return 6
```

```
    elif role == "na":
```

```
        return 7
```

```
    elif role == "research":
```

```
        return 8
```

```
    elif role == "sw":
```

```
        return 9
```

```
job_type_num1 = number_simplifier(job_type_num)
```

```
def senior_simplifier(title):
```

```
    if title == "Senior":
```

```
        return 1
```

```
    else:
```

```
        return 2
```

```
seniority_num = st.radio("Senior role?", options=["Senior", "Not Senior"])
```

```
seniority_num1 = senior_simplifier(seniority_num)
```

```
len_desc = st.number_input('Character Length of the Job Description', step=1.0)
```

```
features = [rating, jobhq, age, num_comp, python_yn, r_yn, aws_yn, spark_yn,  
hadoop_yn,
```

```
            docker_yn, sql_yn, linux_yn, flask_yn, django_yn, tensorflow_yn,  
keras_yn,
```

```
            pytorch_yn, tableau_yn, algo_yn, stats_yn, job_type_num1,  
seniority_num1, len_desc]
```

```
final_features = np.array(features).reshape(1, -1)
```

```
if st.button('Reccomend'):
```

```
    prediction = model.predict(final_features)
```

```
    st.balloons()
```

```
    st.success(f'Your reccomended salary is Rs: {round(prediction[0]*60,3)*100} ')
```

B) References:

- [1]. Anna Palczewska, Jan Palczewski, Richard Marchese Robinson, Daniel Neagu “Interpreting random forest classification models using a feature contribution method”*Integration of Reusable Systems* pp 193-218 (2014)
- [2]. Anand V. Saurkar, Kedar G. Pathare, Shweta A. Gode “An Overview On Web Scraping Techniques And Tools”*International Journal on Future Revolution in Computer Science & Communication Engineering* (2018)
- [3]. Dr. Rajiv Suresh Kumar, Fathima Dilsha, Shilpa A, Sumayya A “Student Placement Prediction Using Support Vector machine Algorithm”*International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering*(2021)
- [4]. ERDİNÇ UZUN “A Novel Web Scraping Approach Using the Additional Information Obtained From Web Pages” *IEEE Access* (Volume: 8) (2020)
- [5]. Hengru Zhang, Fan Min, Shenshen Wang “A random forest approach to model-based recommendation” *JOURNAL OF INFORMATION & COMPUTATIONAL SCIENCE* (2019)
- [6]. Kulkarni, V. & Sinha, P., n.d. Effective Learning and Classification using Random Forest Algorithm. *International Journal of Engineering and Innovative Technology (IJEIT)*.
- [7]. Mitchel, M. T., 1997. *Machine learning*. Burr Ridge, IL: McGraw Hill, 45, 1997
- [8]. Nasteski & Vladimir “An Overview of the Supervised Machine Learning Methods. Faculty of Information and Technology” *Faculty of Information and communication Technologies*
- [9]. Pornthep Khongchai, Pokpong Songmuang “Random forest for salary prediction system to improve students' motivation” *12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)* (2016)
- [10]. Seyed Amir Naghibi, Mojtaba Dolatkordestani, Ashkan Rezaei, Payam Amouzegari, Mostafa Taheri Heravi, Bahareh Kalantar, Biswajeet Pradhan “Application of rotation forest with decision trees as base classifier and a novel ensemble model” *Springer International Publishing* (2014)
- [11]. Suyash Srivastava, Deepanshu Sharma and Priyanka Sharma “Comparing various Machine Learning Techniques for Predicting the Salary Status”*EasyChair Preprint no. 2625*(2020)

- [12]. Sayan Das, Rupashri Barik, Ayush Mukherjee "SALARY PREDICTION USING REGRESSION TECHNIQUES"Industry Interactive Innovations in Science, Engineering & Technology(13SET2K19)(2019)
- [13]. Vrushali Y Kulkarni, Pradeep K Sinha "Effective Learning and Classification using Random Forest Algorithm "International Journal of Engineering and Innovative Technology (IJEIT) Volume 3, Issue 11, May 2014
- [14]. XIANG LI, ZHIJIAN WANG, LIUYANG WANG, RONGLIN HU, AND QUANYIN ZHU "A Multi-Dimensional Context-Aware Recommendation Approach Based on Improved Random Forest Algorithm" IEEE Access (Volume: 6) (2018)
- [15]. Available at: <https://medium.com/machine-learning-101/chapter2-svm-support-vector-machine-theory-f0812effc72>
- [16]. Python <https://www.python.org/>