

# DIC PROJECT PHASE 1

**CSE 4/587B**

## Employability Classification And Salary Estimation

Full Name	UBIT Name	UB Number
Dharma Acha	Dharmaac	50511275
Adarsh Reddy Bandaru	adarshre	50527208
Sri Sahith Reddy Kuncharam	srisahit	50532516

## **Project Statement:**

This project mainly focuses on "Employability Classification of Job Applicants". Its main purpose is to help organizations to check if a candidate is suitable for a job opening. It also helps as a good source for the job applicants who want to know what salary to expect when applying. This solution will be mutually beneficial to both employers and employees.

## **Background of the Problem:**

**Job Market Competition:** Both job applicants and employers are constantly facing challenges in today's competitive job market. A large number of applications always come to employers making it challenging for them to identify the most suitable candidates effectively. This high volume of applications can be a burden for companies and hinder the effectiveness of the hiring process.

**Recruitment Efficiency:** Companies are constantly looking for possibilities to increase the efficiency of their hiring processes. Using this solution as an early screening step will not only save time and resources but also follow the organization's needs.

**Expectations of Applicants:** Applicants can have different salary expectations. So, knowing what salary an applicant can expect from a particular role in a particular country will make things easier for the employee.

## **Objective:**

The primary goal is to obtain the following:

- a) **Candidate Selection based on important features from the dataset:** Our first objective is to create a machine learning model for employers which can pick out the most suitable candidates based on their overall technical skills, total experience, previous job role, etc. With this model, we get the essential criteria that describe whether a candidate is suitable for a particular job role or not.
- b) **Predicting Expected Salary:** Our second aim is to create a predictive model that predicts what salary of a job applicant can expect with his level of skills and experience.

## **Solution:**

We will use predictive and classification machine learning models to obtain expected salaries. We will use algorithms like logistic regression, decision tree, and random forests to classify the employability of a job applicant. Then we implement salary prediction using regression models like linear regression.

## **Cruciality of the Project:**

This project will have a great impact on the human resources department in the field of the hiring process making it more efficient.

In a real competitive job market, organizations generally receive a large number of applications for a single opening. Selecting highly skilled candidates is the main factor for hiring efficiency. It saves valuable time for recruiters and reduces the chances of overlooking suitable candidates for the role.

Salary negotiation is also a critical part of the hiring process. Accurate salary predictions reduce the back-and-forth between job seekers and employers. It ensures that job offers align with market rates and candidate expectations. Thus, improving the overall hiring experience.

### **About the dataset:**

The dataset was taken from Kaggle and it consists of various columns, they are age, education level (EdLevel), status of the employment, gender, mental health, branch (MainBranch), years of coding experience (YearsCode and YearsCodePro), country of residence, previous salary data, work history with a particular technologies ("Have worked with"), computer skills, and status of present employment.

### **Source of the data:**

By considering a period of time, this dataset has been collected from various origins, like job websites, and online applications. Data collection was conducted using standardized methods to maintain balance across various data points. This dataset has a wide variety of industries, job roles, and educational qualifications from different countries.

### **Understanding the data:**

There are “73,462” observations with various features in our dataset.

- Age: age of the applicant, >35 years old or <35 years old (*categorical data*)
- EdLevel: education level of the applicant (Undergraduate, Master, PhD...) (*categorical data*)
- Gender: gender of the applicant, (Man, Woman, or NonBinary) (*categorical data*)
- MainBranch: whether the applicant is a professional developer (*categorical data*)
- YearsCode: how long the applicant has been coding (*numerical data*)
- YearsCodePro: how long the applicant has been coding in a professional context, (*numerical data*)
- PreviousSalary: the applicant's previous job salary (*numerical data*)
- ComputerSkills: number of computer skills known by the applicant (*numerical data*)
- Employed: whether the applicant has been hired or not (*categorical data*)
- Country: Residence of the Employee (*categorical data*)
- Employment: Whether the employee has previous experience or not. (*categorical data*)
- Mental Health: the having good mental health or not (*categorical data*)
- Have worked with: The various languages the employee worked with (*categorical data*)

- ❖ Age
- ❖ EdLevel
- ❖ Employment
- ❖ Gender
- ❖ Mental Health
- ❖ MainBranch
- ❖ YearsCode

- ❖ YearsCodePro
- ❖ Country
- ❖ Previous salary
- ❖ Have worked with
- ❖ Computer skills
- ❖ Employed

## Agenda:

- We try to answer following questions for Employability classification
1. How does the Age and Gender distribution look in the dataset?
  2. Is there any correlation between years of coding experience and salary?
  3. How do mental health issues affect employability?
  4. What are the top most popular technologies that developers have worked with ?
  5. Does having more skills correlate with good employment opportunities?

## Challenges in the data

- Understand the meaning of the columns
- Handling NaN values, null values and outliers
- Implementing multiple visualizations to summarize the information in the dataset and successfully describe the results and trends.

## Adding Noise to the Data:

Here we describe a Python script that focuses on data preprocessing and adding noise to the dataset. The script uses Python libraries like Numpy and Pandas to introduce outliers, null values and data format issues.

### Data Preparation

- `introduce_outliers`: This function introduces outliers in a respective column of the DataFrame by randomly selecting a percentage of rows and replacing their values with an outlier value.
- `introduce_nulls`: This function introduces null (NaN) values in random cells of the DataFrame.
- `corrupt_salary`: This function is used to format the data by randomly selecting a percentage of rows in the 'PreviousSalary' column and appending the string '\$' to the existing values.

### Adding Noise to the Data

- **Introducing Outliers**: Outliers are introduced in two columns, 'YearsCode', 'YearsCodePro', and salary with a 3% outlier rate. This tells the presence of extreme values in these columns.

- **Introducing Null Values:** Null values are introduced across the entire dataset with a 10% null rate.
- **Data Format Issues:** A function to corrupt data format in the 'PreviousSalary'. This function would introduce data format issues to form errors for about 15% of the data.

### Importing necessary libraries:

We use Python libraries such as pandas, seaborn, and matplotlib to perform data visualization and exploration.

**pandas (pd):** We use this library to work with data frames.

**seaborn (sns):** Used it for statistical graphics.

**matplotlib.pyplot (plt):** Also used for statistical graphics.

### **Exploring the data:**

**df.head():** Used to display first n row from the dataset

**df.info()** : provides a summary of the dataset

**df.nunique()** is used to calculate the number of unique values in each column.

**df.isnull().sum(axis=1)** It calculates the number of rows that contain at least one null (missing) value.

### **Understanding the Key Statistics of the data:**

1. Mean
2. Median
3. Mode
4. Minimum and Maximum
5. Standard Deviation

### **Data Cleaning and Preprocessing:**

#### **Step 1: Removing unnecessary columns**

The dataset has an unnecessary column “unnamed 0”, So we dropped it. Removing columns is done to get data clarity and obtain better data analysis.

#### **Step 2: Clean corrupt rows:**

This step involves cleaning and preprocessing the 'PreviousSalary' column. We remove the ‘\$’ symbol is present using string manipulation and replace method.

### Step 3: Removing rows with null values in specified columns:

Rows containing null values in columns ('Employed,' 'ComputerSkills,' and 'HaveWorkedWith') are removed from the DataFrame using the dropna() method. This operation gives a data frame that only has rows where these columns have non-null values. Removing rows with null values in particular columns is a common data preprocessing step. Since replacing the missing values might change and affect the data accuracy, we simply remove them.

### Step 4: Replacing null values:

**Numerical Columns:** The missing values in the columns 'YearsCode,' 'YearsCodePro,' and 'PreviousSalary,' are replaced with the obtained median of each respective column. This is used to preserve the central tendency of the data and reduce the impact of outliers.

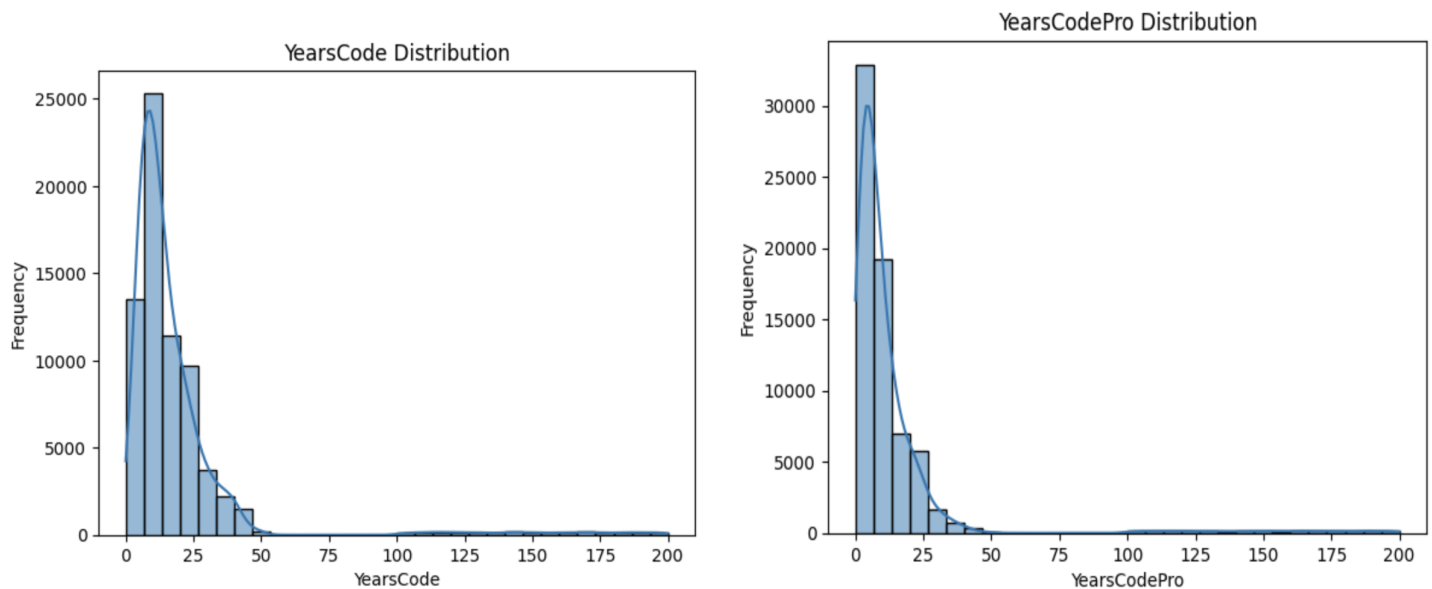
**Categorical Columns:** The columns 'Age,' 'Accessibility,' 'EdLevel,' 'Employment,' 'Gender,' 'MentalHealth,' 'MainBranch,' and 'Country,' are categorical so their missing values are replaced with the mode of each respective column.

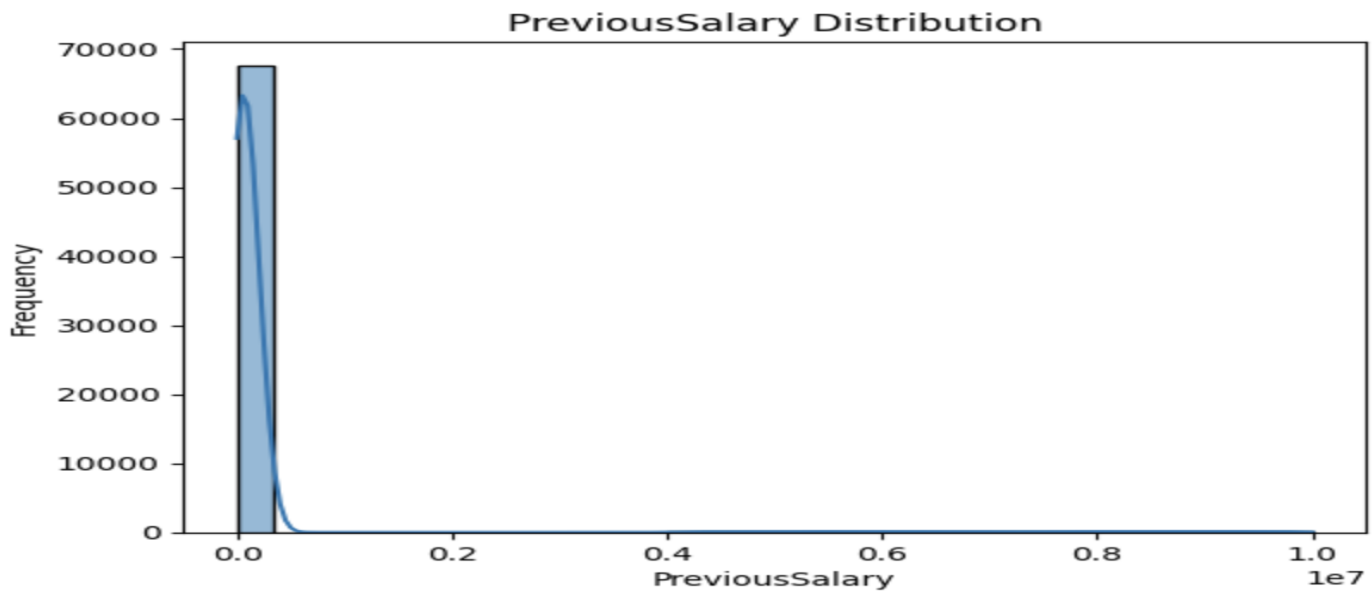
### Step 5: Removing countries with very small sample sizes

The data is preprocessed to get countries which have more than 100 rows under it. This step is done to exclude countries with very few data points which can lead to a model underperforming.

### Step 6: Checking outliers and removing them

**Outlier Visualization:** For each numerical column in the list , a histogram plot is generated using Seaborn's hist plot function. This visualizes the distribution of values and identifies the outliers.





From the above bar graphs, we can analyze the distribution of YearsCode, YearsCodePro and PreviousSalary and see outliers being present.

**Step 7: Outlier Detection using IQR:** After visualization, outliers are identified and removed using the IQR method. For each numerical column, the IQR is calculated as the difference between the third quartile (75th percentile) and the first quartile (25th percentile). The lower and upper limits are defined as 1.5 times the IQR below the first quartile and above the third quartile, respectively.

**Outlier Replacement:** Outliers beyond the limits are replaced with the median value of the respective column. This is done for each row in the data frame, ensuring that values falling outside the IQR are replaced.

### Step 8: Handling String sensitivity in a column

The 'HaveWorkedWith' column in the DataFrame is preprocessed to handle string sensitivity.

Using the str. lower method we can change strings to lowercase. Handling string sensitivity by converting text to lowercase or uppercase is used to avoid duplicates when separating, this makes it easy to perform case-insensitive searches and analyses. The cleaned DataFrame now contains the 'HaveWorkedWith' column with all its text in lowercase letters.

### Step 9: Create a new column for each skill

Set of columns to represent skills that represent the 'HaveWorkedWith' column, Here we split the semicolon-separated skill names into a list and update the skills set with these skills. This step is to get a unique set of all skills given in the dataset. We then create binary columns for all these skills and fill in 1 or 0 based on the row value.

### Step 10: Scaling Numerical columns using min-max scalar

The dataset consists of the following numerical columns ('YearsCode,' 'YearsCodePro,' 'PreviousSalary,' and 'ComputerSkills'). We scale these columns from 0 to 1 using a min-max scaler. This will be quite useful for

increasing the performance of an algorithm that uses gradient descent like logistic regression, linear regression and neural networks.

### Step 11: Encoding categorical data

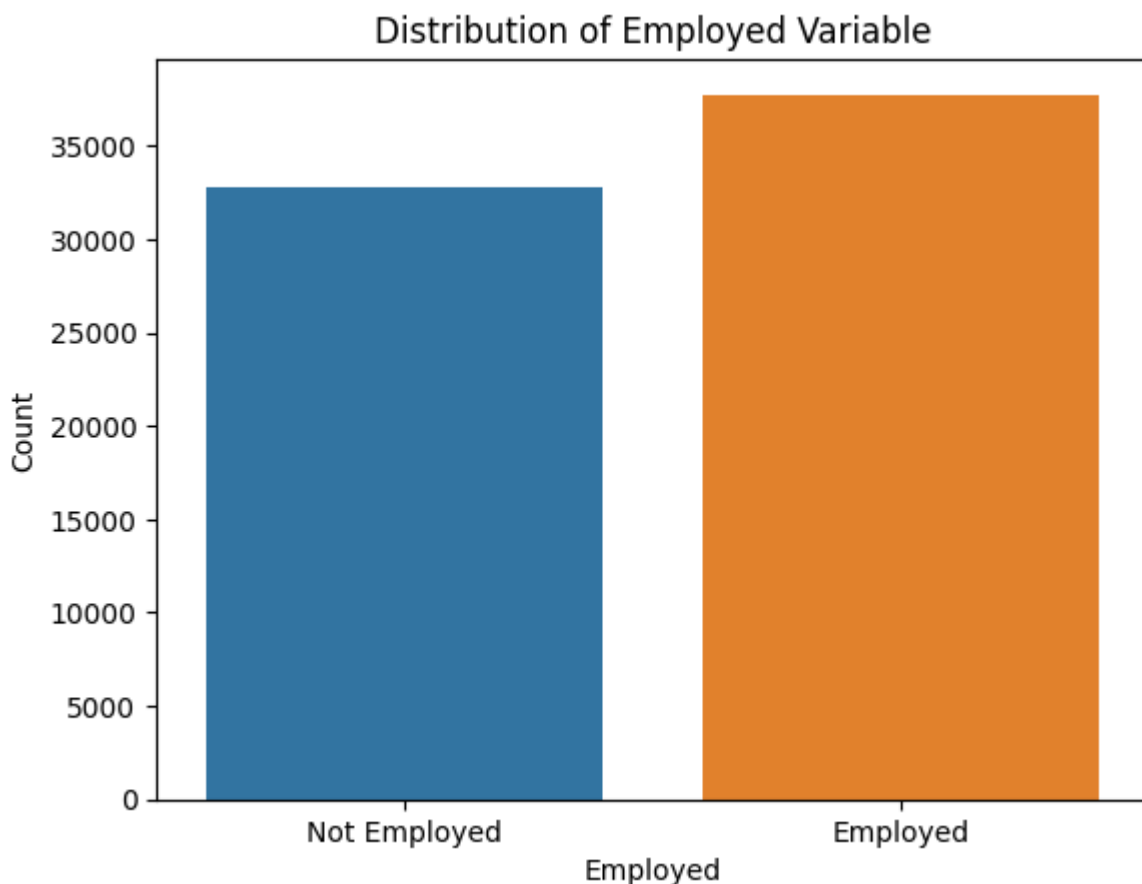
This is obtained by using `pd.get_dummies()` on the df. One-hot encoding helps in converting categorical variables into binary columns,

Our dataset has the following categorical values, such as 'Age,' 'Accessibility,' 'EdLevel,' 'Employment,' 'Gender,' 'MentalHealth,' 'MainBranch,' 'Country,' and 'Employed,' and these column values are encoded using one-hot encoding method. The resulting data frame now contains binary values for each column. This analysis or modelling.

### Exploratory Data Analysis:

#### 1. Distribution of Target variable “Employed”

We checked the distribution of our target variable “Employed” to detect any data imbalances and to check if we have enough data in both classes to make an efficient model.



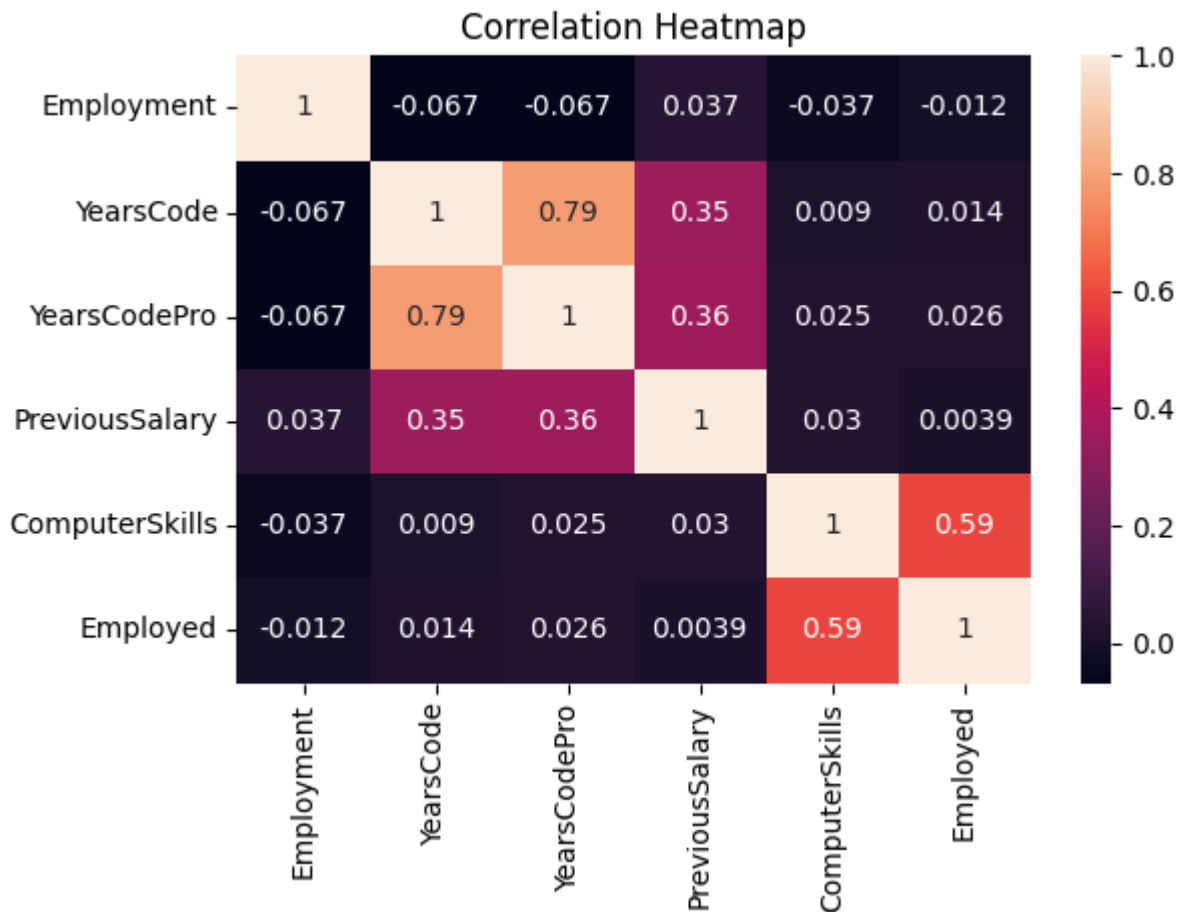


## Observation:

From the above graph, we can say that the target variable is fairly balanced between employed and not employed with a split of 46.5% - 53.5%. This will simplify the model training without using techniques to handle imbalanced datasets.

## 2. Correlation heatmap

To understand the dependence of one column on another and to find any linear patterns, we use a correlation heatmap.



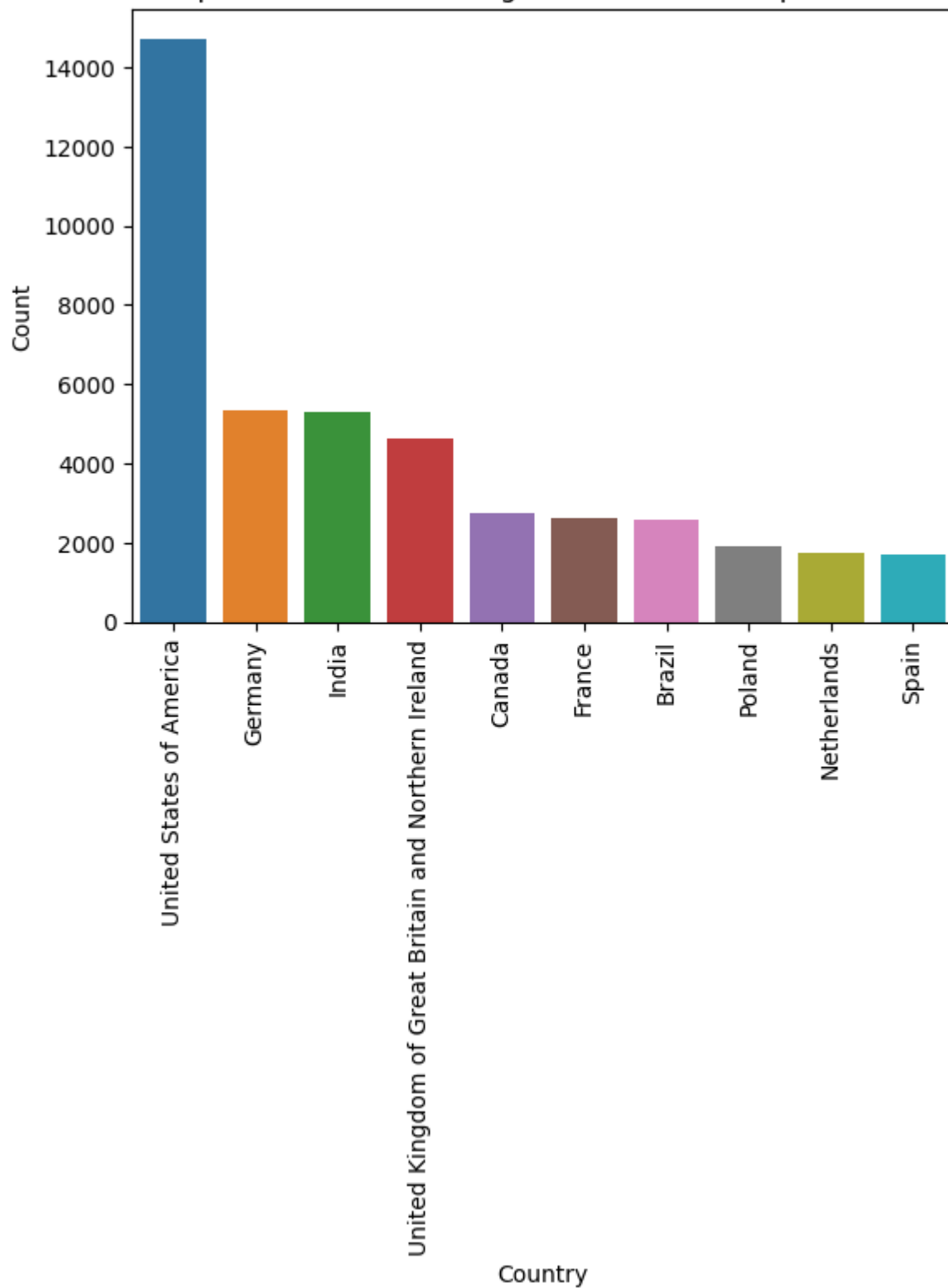
In the above correlation heatmap we can see that there are few correlations. Black color indicates a weak correlation and orange indicates a strong correlation.

We see a moderate correlation between computer skills and employed columns. There is also a strong correlation between YearsCode and YearsCodePro columns which makes sense. With the help of this map, we can rule out the question “Does years of coding experience have a positive impact on employability?”

## 3. Top 10 countries with the highest number of respondents.

To analyze the highest number of job applicants applying for a job position from different countries. This can be achieved by plotting a bar graph.

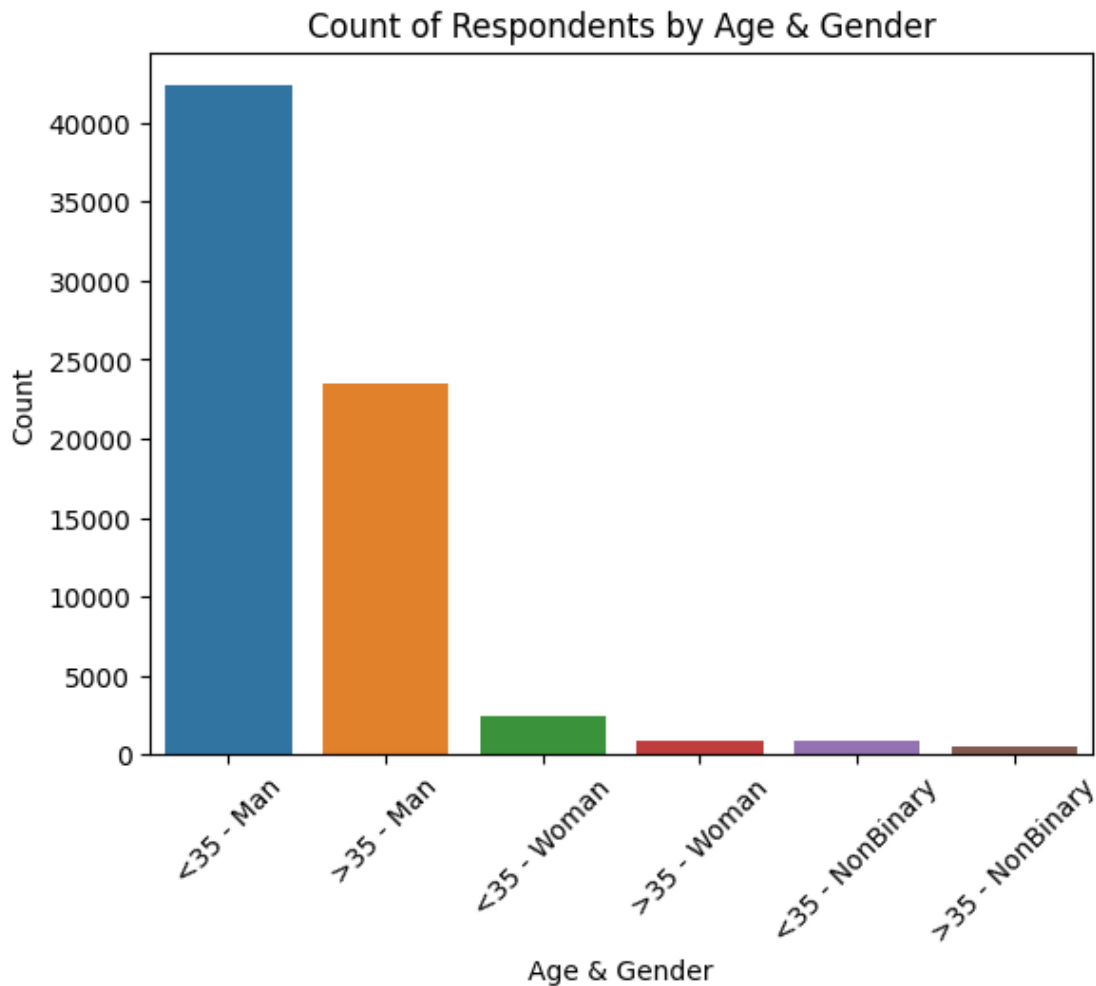
Top 10 Countries with highest number of respondents



From the above visualization, we conclude that the United States of America have a significant number of respondents, after the US, Germany and India have a more or less equal number of applicants. On the other hand, Countries Canada, France, and Brazil recorded half of the applicants when compared with India.

#### 4. Count of Respondents by Age and Gender

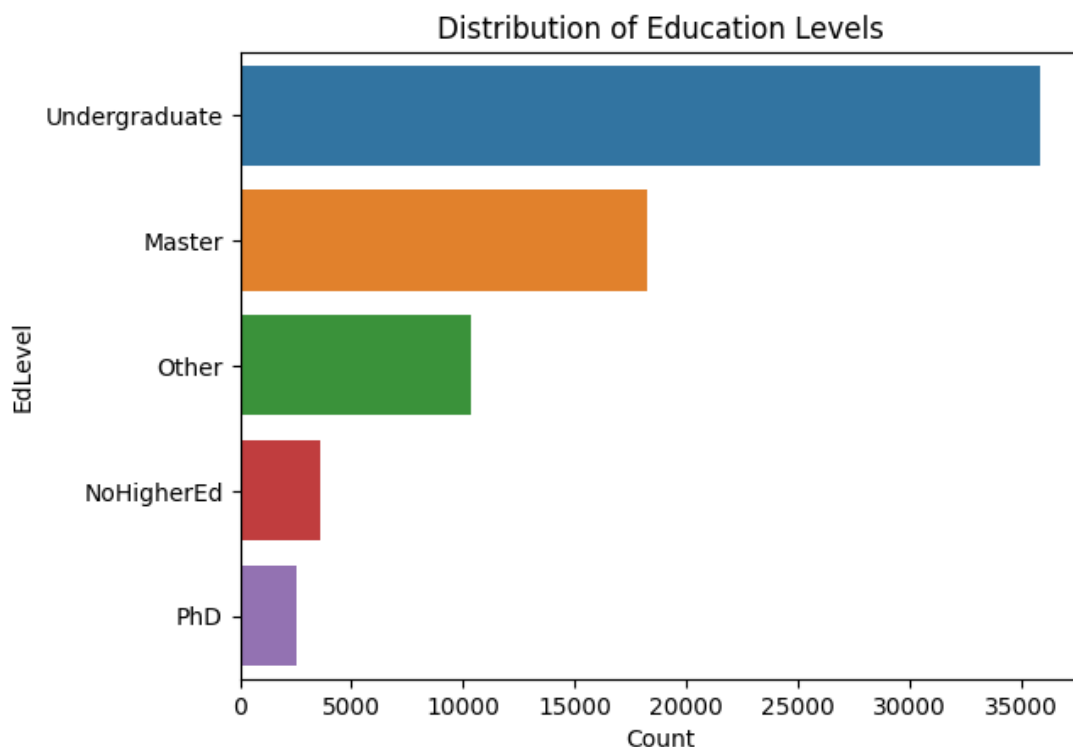
The graph describes the total number of respondents among different genders and ages.



From the above visualization, we can depict that the number of respondents of age less than 35 and gender male are the highest. The number of female respondents is significantly less compared to males and the trend drops even more for non-binary. We also observe that respondents from the age group of less than 35 are higher than the number of respondents aged >35.

#### 5. Explore Education Level.

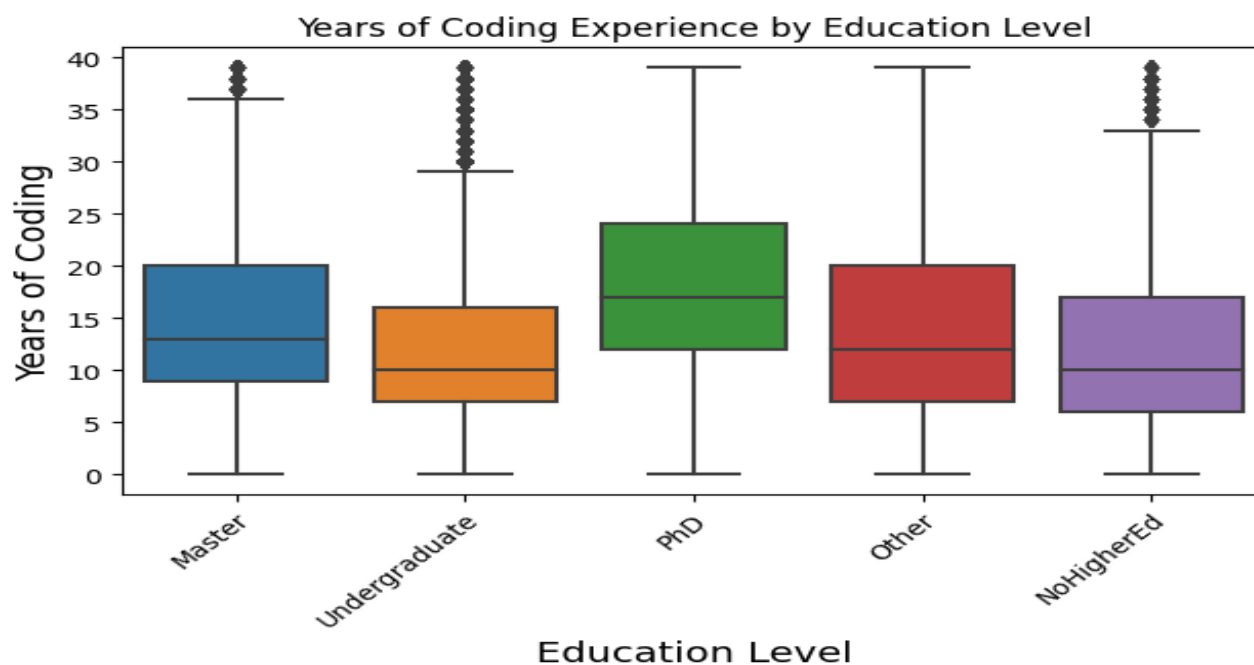
To see the distribution of respondents from each education level, we are using the below visualization.



From the above graph, we see that candidates with undergraduate degrees are more in number which is 35000, whereas PhD are very low in number close to 2000 numbers. And the applicants who have other degrees have a count of 10000, While the respondents who do not have any higher degree settled at 5000.

## 6. Years of Coding Experience by Education Level.

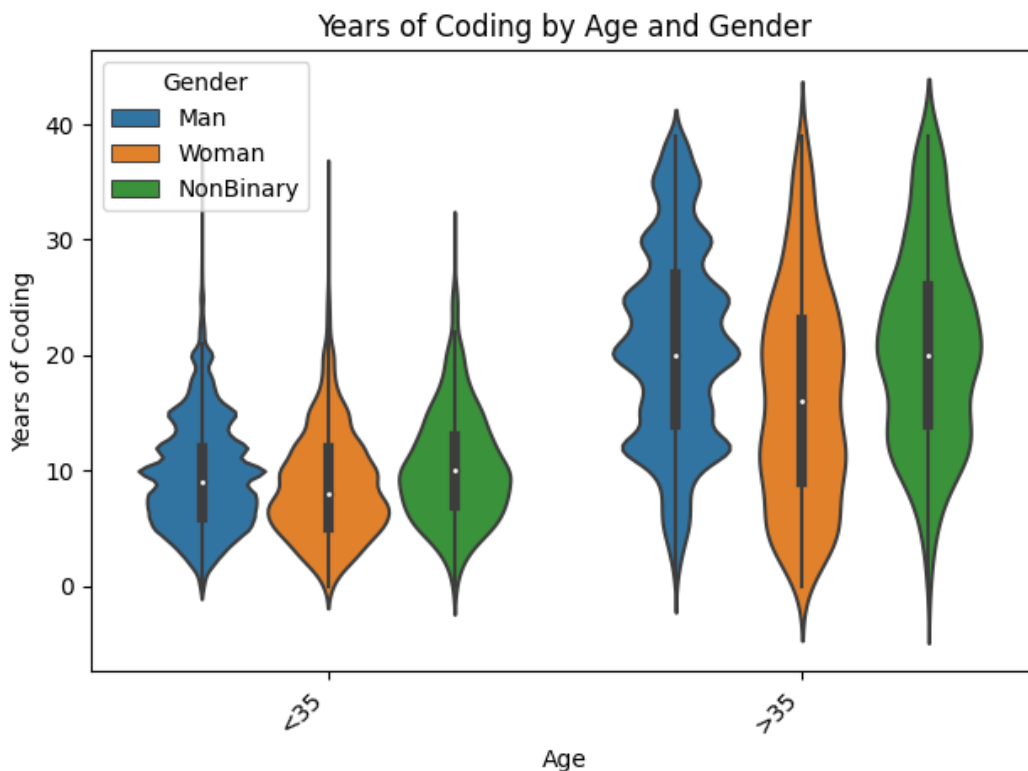
The box plot describes how many years of coding experience candidates have based on their education level.



From the boxplot, we can infer that there is a wide range of coding experience across all education levels. Although there are outliers which indicate that these people may have started coding very early. We also observe that the code experience is going higher as the education level is higher. Candidates with NoHigherEd have coding experience from 6 to 18 years and those with undergraduate degree fall in 8-17 years of coding experience.

## 7. Years of Coding by Age and Gender

The violin plot represents how many years of coding experience candidates have based on their Age and Gender.

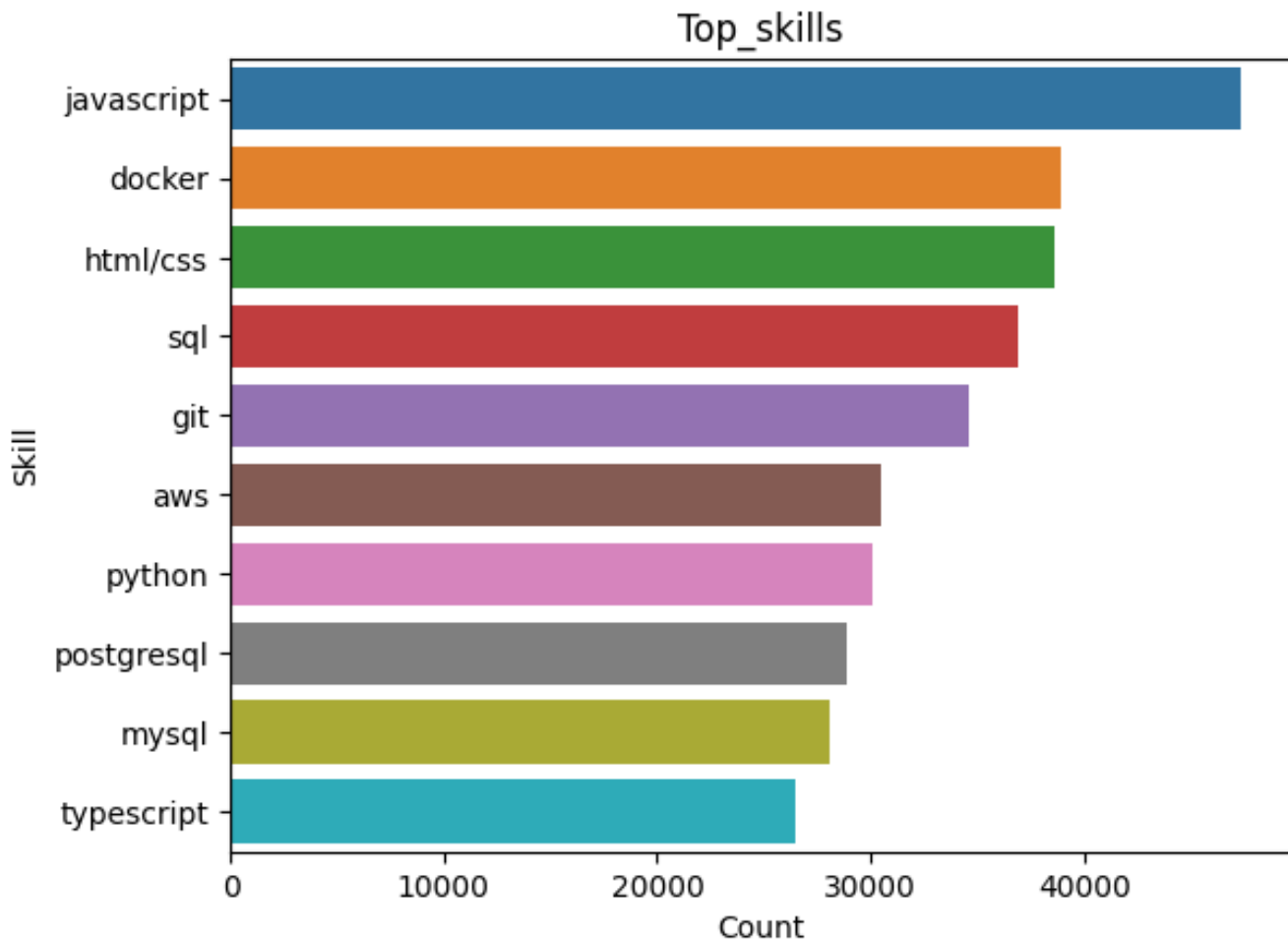


The graph states younger age groups have a higher distribution of coding experience, mostly at the lower end, suggesting that they are in the early stages of their careers. In some age groups, males tend to have a slightly wider distribution.

The male category has more candidates with coding experience of 10 -12 years, Whereas Female category candidates fall in 6-7 years of experience. For candidates with ages greater than 35, the spread of data is greater. Non-binary gender has also a median coding experience of 20 years.

## 8. Top Skills

To determine the skills that the most candidates have.

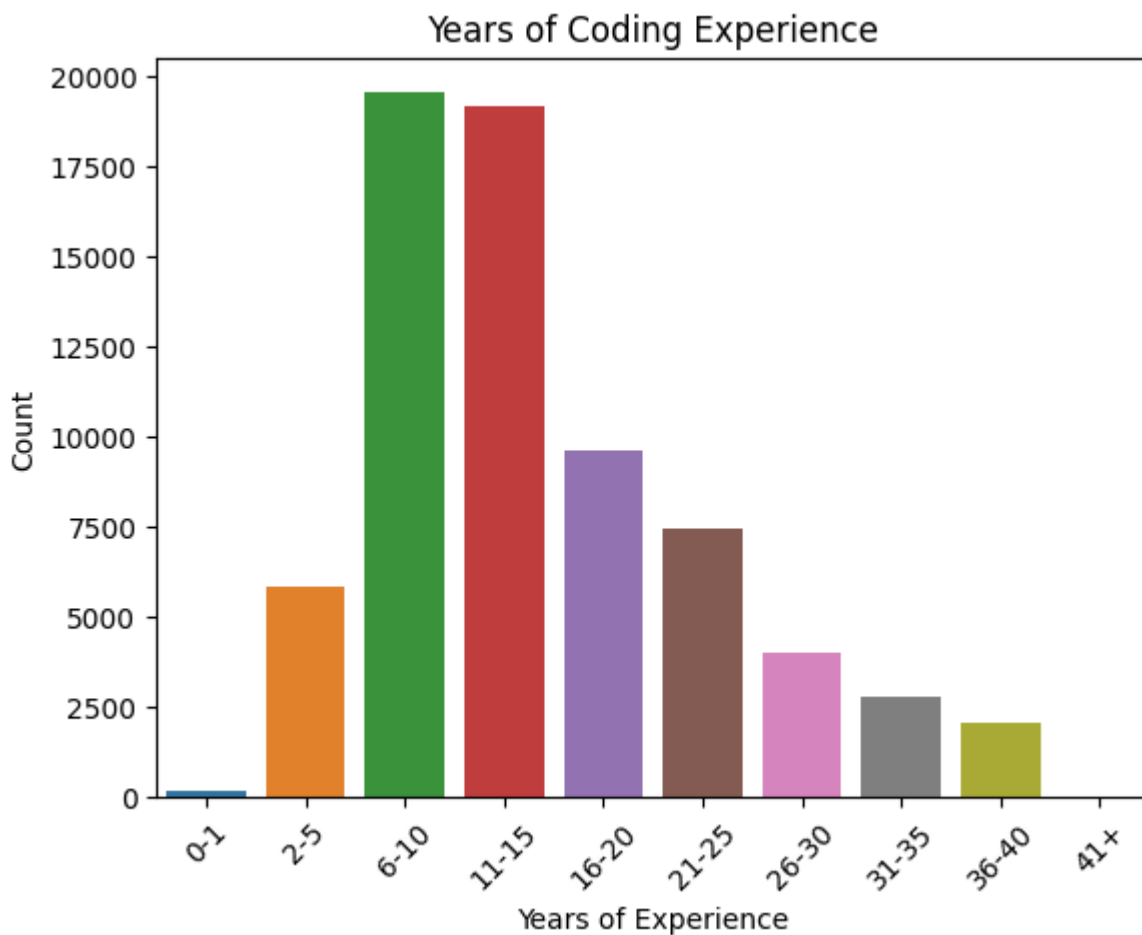


The above visual allows us to determine the top skills that the candidates possess, we see that javascript has been the most occurring skill with more than 40000 candidates knowing how to write code in javascript.

Next place, we see docker and html/css being the most popular skills known by candidates. In third place, we see SQL followed by git, AWS, and python.

## 9. Years of Coding Experience

Grouping candidates based on how many years of coding experience they have.



We see a significant number of applicants who have coding experience of 6-15 years. And also a significant drop in counts as the experience increases indicating people retiring which is a common trend. Around 6-10 years of coding experience is more in number than compared to any other age category.

Around 6-10 years of coding experience are more in number than compared to any other age category.

Next comes 11 - 15 years of experience. In third place we have 16 - 20 years of experience. We see that the 41+ years category is zero in number. And with 0 - 1 year experience are very less compared to all other categories.

## 10. Effect of coding experience on salary

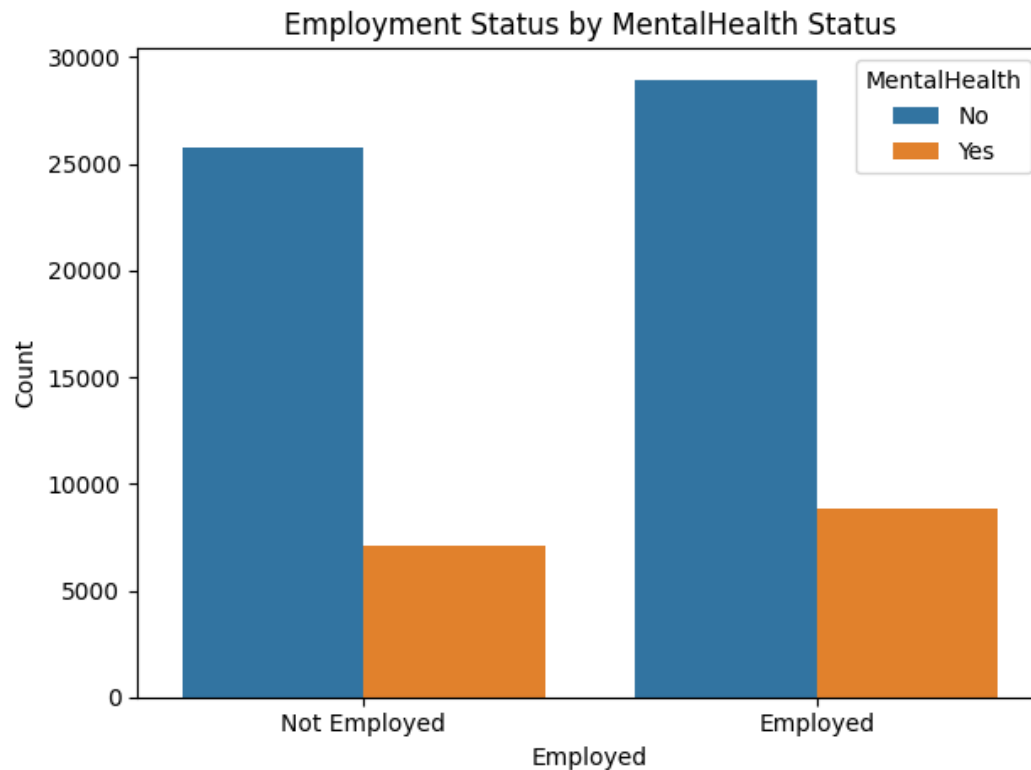
To see how coding experience has effects on previous salary, we are using this visual.



The above plot shows no linear relation between years of coding experience and previous salaries as the graph does not show any patterns.

### 11. Employment Status by MentalHealth Status.

How mental health status is affecting employment status.

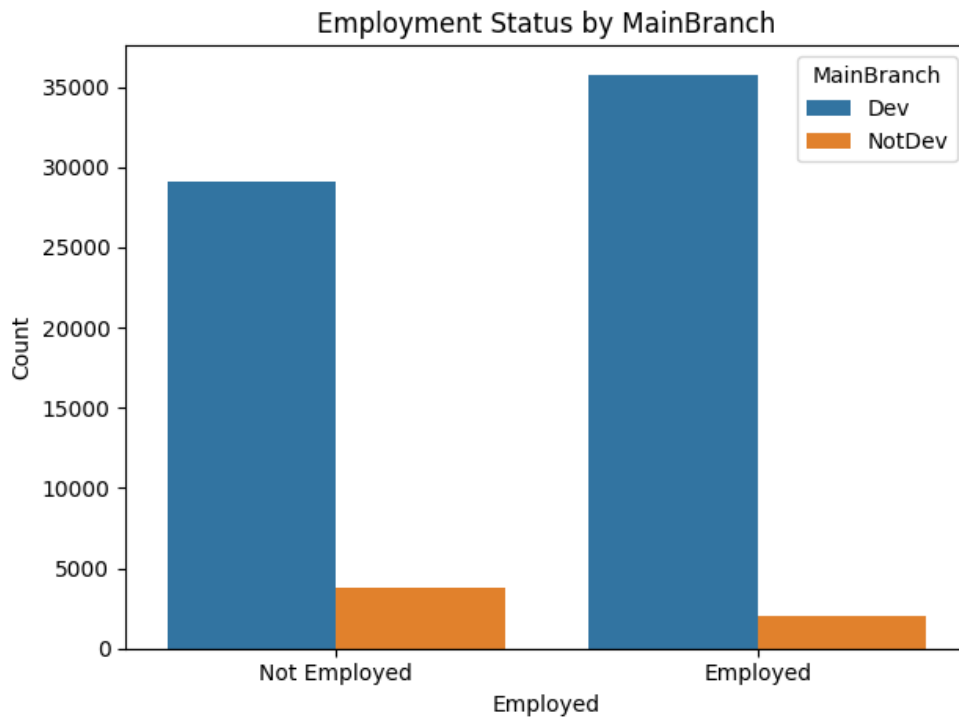




The graph shows consistent employment among the different mental conditions, we see that both employed and not employed have consistent no mental health count as well as consistent mental health count.

## 12. Employment Status by MainBranch

To view the employment status based on the Developer and Non-developer roles.



We see that developers tend to have more employment compared to non-developers. The same goes with non-employed candidates too, developers tend to do more, and based on the above visualization we can depict that.

## Exploratory Data Analysis Conclusion:

After trying to observe employability factors and trends, the dataset reveals a balanced distribution between employed and non-employed individuals. We see some correlations between YearsCode and YearsCodePro, and a moderate one between computer skills and employed status. The demographic explorations state USA has the most number of respondents. Educational breakdowns denote a majority of undergraduates and show a strong relation with years of coding experience. When checking for top skills among the applicants, JavaScript came out at the top. Lastly, we found no connection between years of coding experience and previous salaries and found consistent employment trends with mental health statuses, and previous job roles. We will use the following findings to make a machine-learning model that works as an effective hiring tool.

## References:

1. <https://www.kaggle.com/datasets/ayushtankha/70k-job-applicants-data-human-resource>
2. <https://www.itl.nist.gov/div898/handbook/eda/section1/eda11.htm>

# DIC PROJECT PHASE 2

**CSE 4/587B**

## Employability Classification And Salary Estimation

Full Name	UBIT Name	UB Number
Dharma Acha	Dharmaac	50511275
Adarsh Reddy Bandaru	adarshre	50527208
Sri Sahith Reddy Kuncharam	srisahit	50532516

## **Project Statement:**

This project mainly focuses on "Employability Classification of Job Applicants". Its main purpose is to help organizations to check if a candidate is suitable for a job opening. It also helps as a good source for the job applicants who want to know what salary to expect when applying. This solution will be mutually beneficial to both employers and employees.

## **Background of the Problem:**

**Job Market Competition:** Both job applicants and employers are constantly facing challenges in today's competitive job market. A large number of applications always come to employers making it challenging for them to identify the most suitable candidates effectively. This high volume of applications can be a burden for companies and hinder the effectiveness of the hiring process.

**Recruitment Efficiency:** Companies are constantly looking for possibilities to increase the efficiency of their hiring processes. Using this solution as an early screening step will not only save time and resources but also follow the organization's needs.

**Expectations of Applicants:** Applicants can have different salary expectations. So, knowing what salary an applicant can expect from a particular role in a particular country will make things easier for the employee.

## **Objective:**

The primary goal is to obtain the following:

- a) **Candidate Selection based on important features from the dataset:** Our first objective is to create a machine learning model for employers which can pick out the most suitable candidates based on their overall technical skills, total experience, previous job role, etc. With this model, we get the essential criteria that describe whether a candidate is suitable for a particular job role or not.
- b) **Predicting Expected Salary:** Our second aim is to create a predictive model that predicts what salary of a job applicant can expect with his level of skills and experience.

## **Solution:**

We will use predictive and classification machine learning models to obtain expected salaries. We will use algorithms like logistic regression, decision tree, and random forests to classify the employability of a job applicant. Then we implement salary prediction using regression models like linear regression.

## **Cruciality of the Project:**

This project will have a great impact on the human resources department in the field of the hiring process making it more efficient.

In a real competitive job market, organizations generally receive a large number of applications for a single opening. Selecting highly skilled candidates is the main factor for hiring efficiency. It saves valuable time for recruiters and reduces the chances of overlooking suitable candidates for the role.

Salary negotiation is also a critical part of the hiring process. Accurate salary predictions reduce the back-and-forth between job seekers and employers. It ensures that job offers align with market rates and candidate expectations. Thus, improving the overall hiring experience.

## **About the dataset:**

The dataset was taken from Kaggle and it consists of various columns, they are age, education level (EdLevel), status of the employment, gender, mental health, branch (MainBranch), years of coding experience (YearsCode and YearsCodePro), country of residence, previous salary data, work history with a particular technologies ("Have worked with"), computer skills, and status of present employment.

## **Source of the data:**

By considering a period of time, this dataset has been collected from various origins, like job websites, and online applications. Data collection was conducted using standardized methods to maintain balance across various data points. This dataset has a wide variety of industries, job roles, and educational qualifications from different countries.

## **Introduction:**

In order to gain insights from data, we have applied six different algorithms which does classification and prediction tasks, here below mentioned algorithms we have used to classify and predict.

## **Classification Algorithms:**

Decision Tree  
K-Nearest Tree  
Naive Bayes  
Random Forest  
XGBoost

## **Prediction Algorithm:**

Linear Regression

Machine learning offers a variety of algorithms with its unique strengths and applications. In the context of employability classification and salary prediction, several algorithms stand out for their effectiveness and several algorithms stand out for their effectiveness and utility. This analysis examines the application of various machine learning models mentioned above in the scenario of predicting job applicants' suitability and salary expectations.

The primary focus is on understanding how these models can benefit companies in making better hiring decisions. By examining each model's performance and application to real-world hiring scenarios, we gain insights into their roles in the decision-making process. The analysis not only says about the strengths and weaknesses of each algorithm but also sheds light on the subtle differences of their application in a domain as critical as employment and salary determination.

From the highly interpretable Decision Trees to the efficient Naive Bayes, and from the ensemble skill of Random Forest and XGBoost to the simplicity of Linear Regression in salary prediction, each model presents a unique approach to tackling the complexities of employability classification. The goal is to explore how these models contribute to a more efficient hiring process, leveraging their individual capabilities to enhance decision-making in human resources.

## **Decision Tree :**

Decision tree classifier is one of the most popular machine learning algorithms which can be used for both classification and regression tasks. Basically it is a supervised machine learning algorithm which is easy to understand and interpret.

The problem statement mainly deals with Employability Classification and salary prediction. Since Decision tree classifier is widely used for classification purposes. It serves various purposes for our problem statement.

The primary purpose of using a decision tree in this project is a classification model that can help companies to select the most suitable candidates based on skills, experience. The main goal of the decision tree is to predict whether a new job applicant is good to be hired or not.

### **Why Decision Tree is chosen:**

This classifier is highly interpretable. We used this classifier because it provides a clear representation of how the model makes classifications. This classification is very crucial for the organizations in the circumstances where the institutions want to know if the job applicant is suitable for a role or not.

With the help of this classifier we can easily justify hiring decisions to both job applicants and other employers as they can visually trace the decision path.

Decision trees provide more insights where features have most influence in determining employability such as qualification, skills, previous employability.

Decision trees generally handle binary classification problems. This classification best suits for classifying applicants into suitable for a role or not.

### **Expectations from the Decision Tree**

We will be having a predicting model that can classify job applicants into suitable roles or not based on their attributes.

This classifier makes us to understand which qualifications, skills plays a prominent role in selection which helps companies to understand the main factors and also we can expect proper systematic ways to identify the most suitable job applicants based on attributes

### **How we applied to our problem statement**

We created a Decision Tree classifier with a random state which helps to model to predict the same values every time. The decision tree model is trained on the provided training data. So during this process the model tries to learn decision rules based on features in the dataset which is trained.

After training the model tries to predict the test data. It applies to learned decision rules to predict the testing data that classifies job applicants whether they are suitable or not.

We also calculated various evaluation metrics such as accuracy, precision, and recall to get the model's preference. These metrics help to determine how good Decision Tree is at making accurate predictions and separate candidates from unsuitable ones.

## What we learned from the process

Through this process we achieved various insights. We gained some experience in interpreting decision tree models, this is highly important for maintaining transparency in the hiring process. We also learnt how to improve the hiring process that makes it highly efficient and fair in a competitive world.

We also plotted various graphs to understand the performance of the model

### Performance Graphs:

Below are the performance metrics we obtained.

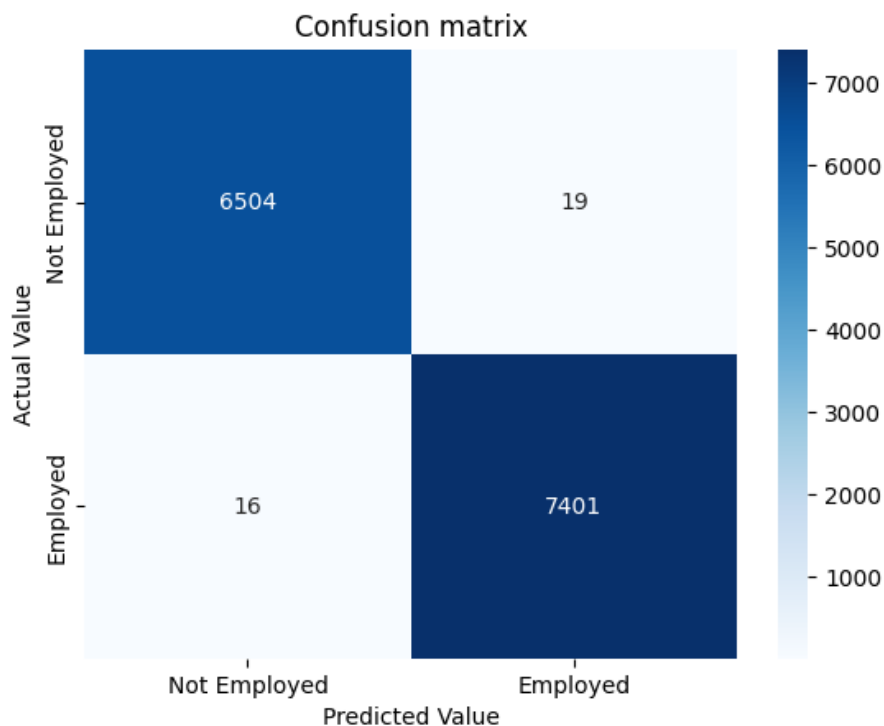
Accuracy: 99%

Precision: 99%

Recall: 99%

### Confusion matrix:

This plot represents the number of true positives, true negatives, false positives and false negatives predicted by the model.



From the graph we can infer that model as predicted 7401 candidates as employed who are already employed and 6504 as not employed

TP: 7401

TN: 6504

FP: 19

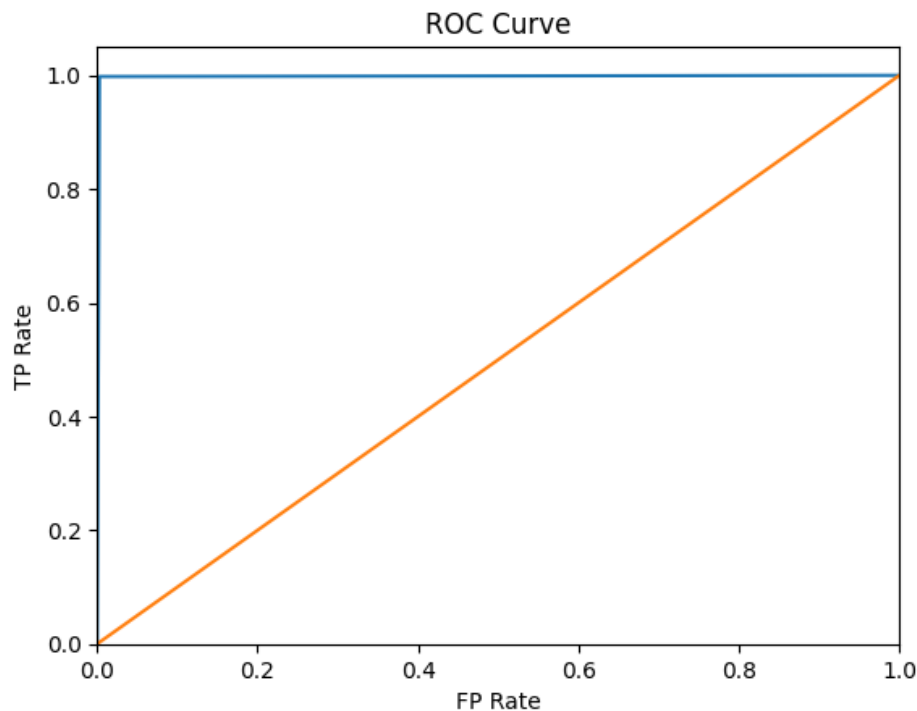
FN: 16

### ROC Curve:

(ROC) stands for Receiver Operating Characteristics. This curve shows performance of binary classification models at various classification thresholds. The graph is plotted with False positive rate against True positive rate. In this graph each point on the ROC curve represents performance of the model at a particular threshold. The Area under Curve (AUC) says overall performance of the model and this value ranges from 0 to 1.

From the graph it is clear that  $AUC \sim 1$  so the model perfectly knows the positive and negative instances. Basically the model whose AUC is closest to 1 will make better predictions.

ROC- Curve area– 0.99

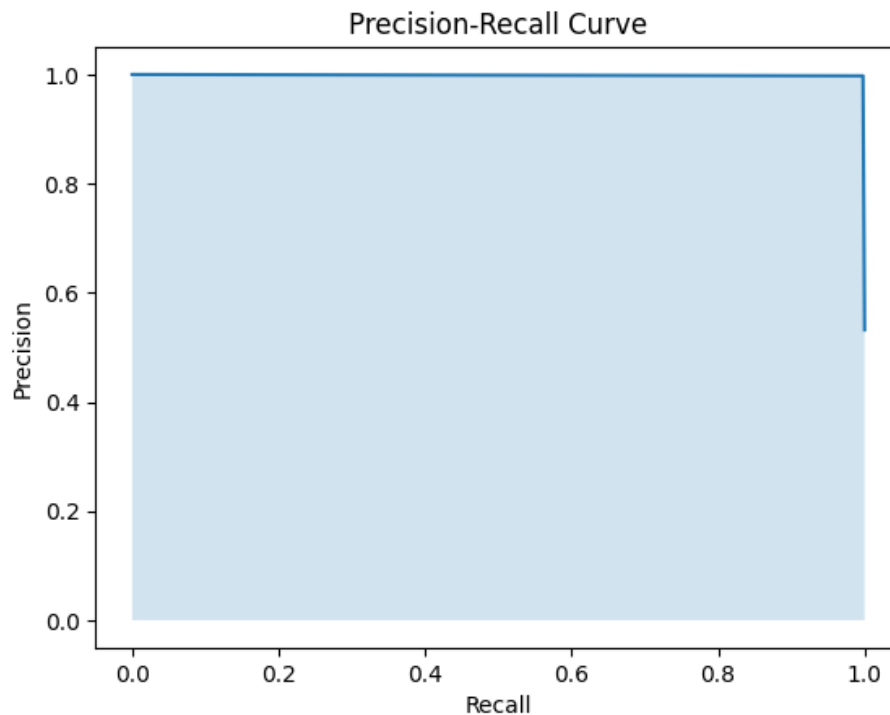


### Precision-Recall Curve:

This curve is plotted by considering Recall and Precision. Similar to the ROC curve, the Precision-Recall curve is described as the area under the curve (AUC-PR). The higher AUC-PR value indicates the model will predict perfectly. This AUC-PR also ranges from 0 to 1.

From the below graph when we calculate the area under AUC-PR curve we obtain as approximately equals to 1. Hence it is clear that our model which is using a decision tree classifier makes better predictions.

Precision-Recall curve area: 0.998



### **K-Nearest Neighbors:**

KNN is the simple machine Learning algorithm mainly used for classification purposes. The basic concept of the KNN is to predict the label of the new data point by considering labels of the K nearest neighbors. In detail if the majority of the already existing data points belong to class then a new point which has these data points as nearest neighbors also gets the same class name.

### **Why KNN is chosen:**

This KNN algorithm best suits our classification task, So we used KNN algorithm as well for our problem statement. And also it works well for both numerical and categorical features. Our dataset has different features like age, education level, coding experience etc; So this algorithm easily handles these mixed features.

One main reason for selecting KNN is to reduce the filtering time of the job applicants, The KNN only observes K-nearest datapoints. In the employability context the applicants with similar profiles which includes technical skills, experience and other are more likely to have employment opportunities.

### **Expectations from the KNN:**

This algorithm is expected to help in selection of candidates by identifying similar profiles who have past experience and good technical skills. This algorithm classifies new candidates by observing learned patterns in training data.

### **How we applied to our problem statement**

We initialized the model with KneighborsClassifier with  $k=5$  neighbors. And after the model is trained on training data using fit method. The predictions are made on the test data using a trained model. And also



predicted values of the positive class are calculated. The inbuilt function `model_conf.ravel()` helps in calculating True positive, False positive, False Negative, , False Negative.

We also calculated various evaluation metrics such as accuracy, precision, and recall to get the model's preference. These metrics help to determine how good KNN is at making accurate predictions and separate candidates from unsuitable ones.

### What we learned from the process:

We learned that KNN provides predictions based on similarity of data points. This is used for understanding factors influencing the employability. We also understood the concept of data similarity plays a vital role and considers data points in the feature space, and the importance of that gives similar outcomes.

We also observed the influence of the number of neighbors chosen is the key factor on model performance. Adjusting this parameter changes the model's capability to learn patterns in the trained data.

We plotted various performance graphs to understand how good the model is predicting

### Performance Graphs:

Below are the performance metrics we obtained

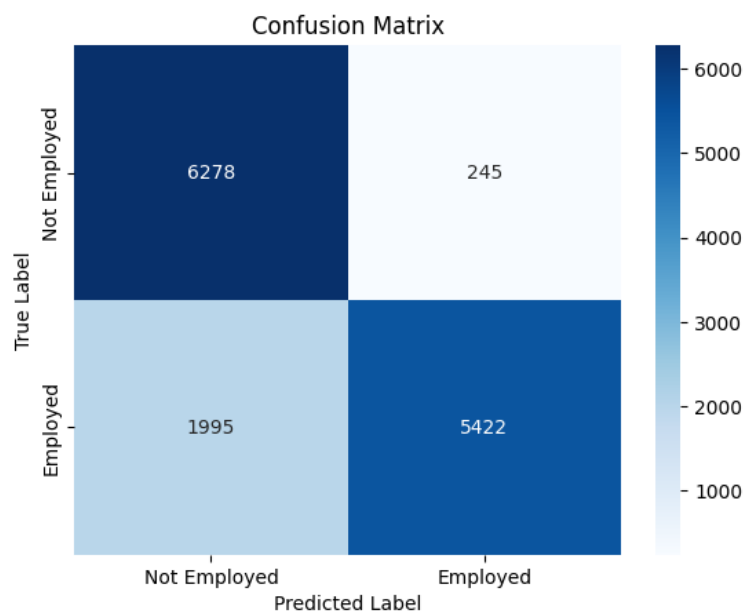
Accuracy: 83.93%

Precision: 95.67%

Recall: 73.10%

### Confusion matrix:

This plot represents the number of true positives, true negatives, false positives and false negatives predicted by the model.



From the plot we can see that KNN model predicted 5422 candidates correctly as employed and 6278 candidates as not employed which are True positives and True Negatives.

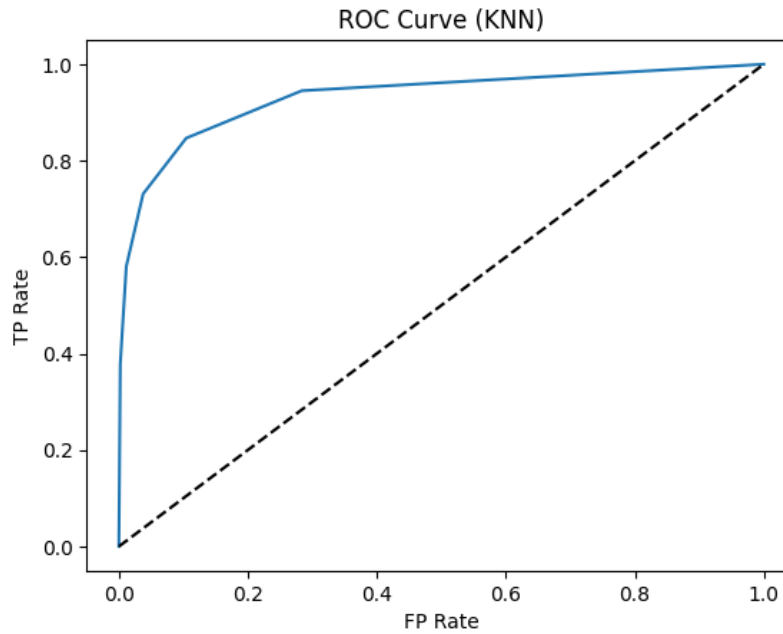
TP: 5422

TN: 6278

FP: 245

FN: 1995

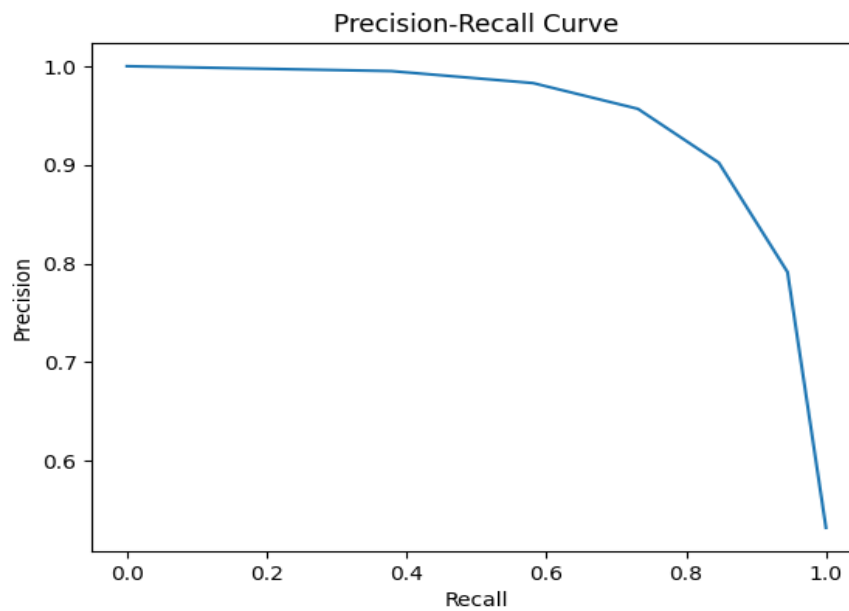
### ROC Curve:



From the graph we can say that the model is performing decently in classifying the positive and negative cases. But it might be missing more true positives.

ROC- Curve area– 0.93

### Precision-Recall Curve:



From the above graph when we calculate the area under AUC-PR curve we obtain as 0.95. So this model is wrongly predicting fewer instances. This can be known from looking into the confusion matrix.

Precision-Recall curve area: 0.95

## **Naive Bayes:**

Unlike decision trees and KNN, naive bayes is a probabilistic classifier and it is very effective against complex classification tasks. It classifies data based on how likely an event can happen.

### **Why Naive Bayes was chosen:**

The main reason for trying out naive bayes is due to its efficiency with large data with a lot of features like in our case. We have a mix of numerical and categorical columns which makes naive bayes a good choice to include as it can manage mixed data types well.

### **Expectation from Naive Bayes:**

We are expecting better performance in terms of how well the model categorizes between employed or not employed based on their profiles.

### **Application to our problem statement:**

Here we used the scikit-learn library GaussianNB model which is a variant of Naive Bayes. We trained the model on a training set and used the probabilistic approach to classify the test data samples. We also derived the probabilities of each class to evaluate the model's overall confidence towards its predictions.

### **Learnings from process:**

We were able to get a decent performance but not as good as the decision tree and KNN. This might be due to the number of independent relationships present in the data.

### **Performance Analysis:**

Below are the key performance metrics obtained:

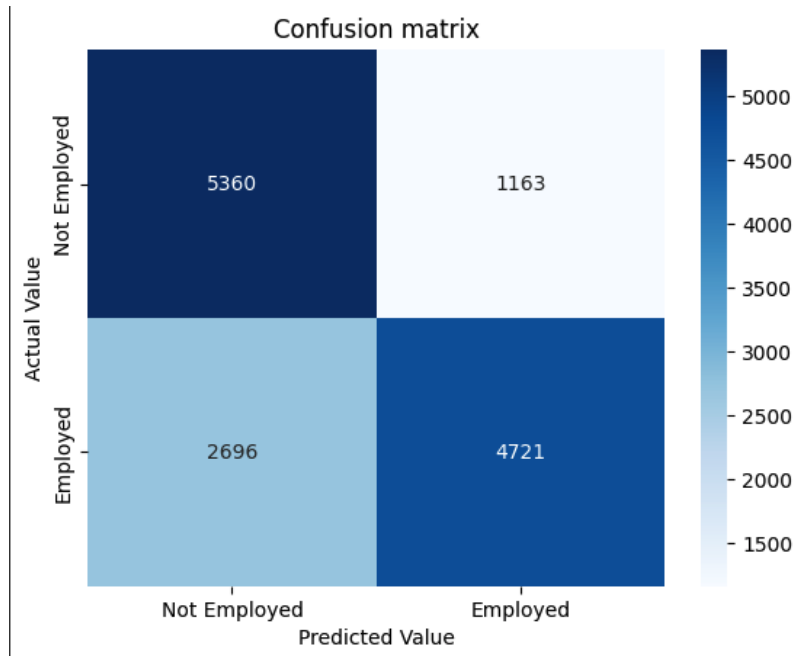
Accuracy: 72.3%

Precision: 80.2%

Recall: 63.6%

### **Confusion Matrix:**

The confusion matrix revealed that the model correctly predicted 5,360 candidates as 'Not Employed' and 4,721 as 'Employed', while it incorrectly classified 1,163 as 'Employed' and 2,696 as 'Not Employed'.



TN: 5,360

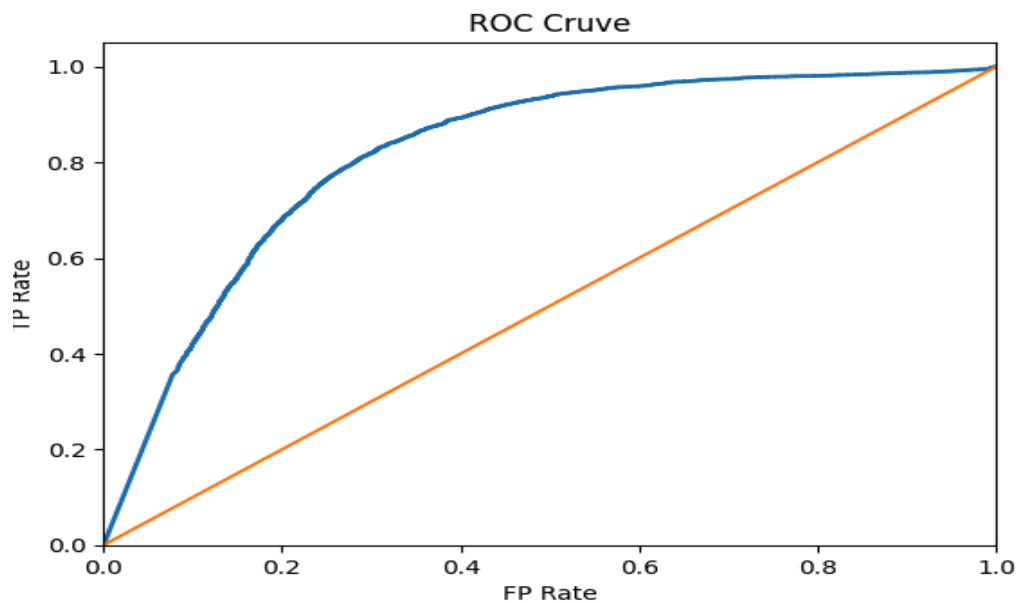
TP: 4,721

FP: 1,163

FN: 2,696

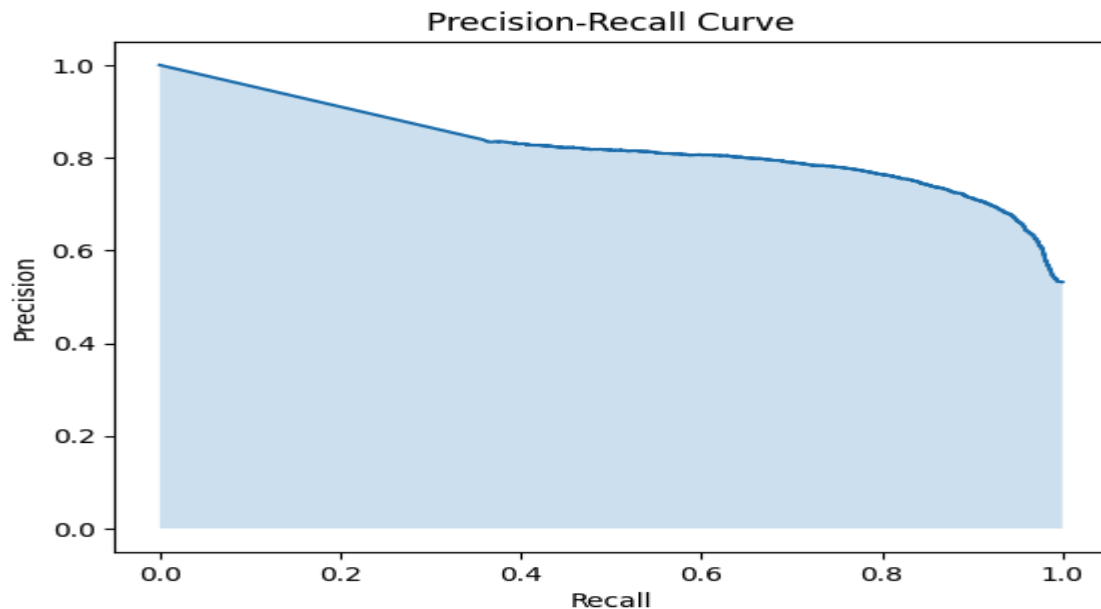
### ROC Curve:

The ROC curve, with an area of 0.82, tells us that the model gives a good performance in identifying between the 'Employed' and 'Not Employed' classes. However, the false positive rate isn't very good.



### **Precision-Recall Curve:**

The auc for the Precision-Recall curve below is 0.82, which tells us that the model is decent in predicting the 'Employed' class, and as discussed it is prediction to many employed and caused more false positives.



### **Random Forest:**

Random forest uses multiple decision trees during training and gives that class that is the model of the classes we got from individual trees. This Algorithm is very effective against overfitting and good for large datasets.

#### **Why Random Forest was chosen:**

We saw a very good performance from the decision tree algo and thought the use of ensemble methods used by random forest might give even better results. Random forest also has the capability of recognizing complex patterns which is also a reason for choosing random forest classifier..

#### **Expectations from Random Forest:**

From using random forest, we are trying to achieve more accuracy than a decision tree algorithm and enhance the decision making process. We also wanted to observe if there is any overfitting in earlier models which is generally reduced by random forest.

#### **Application to Our Problem Statement:**

Just like the above algorithms, we trained the model on training data and tested it on test data. We use scikit-learn libraries to achieve this. We also used it to evaluate the model on various performance measurement metrics.

#### **Learnings from the Process:**

Although the random forest did not give better results than the decision tree model, it can be effective in classifying new data points it encounters. It also did with high-dimensional data and its capability to provide reliable and interpretable results.

## Performance Analysis:

Key performance metrics obtained are:

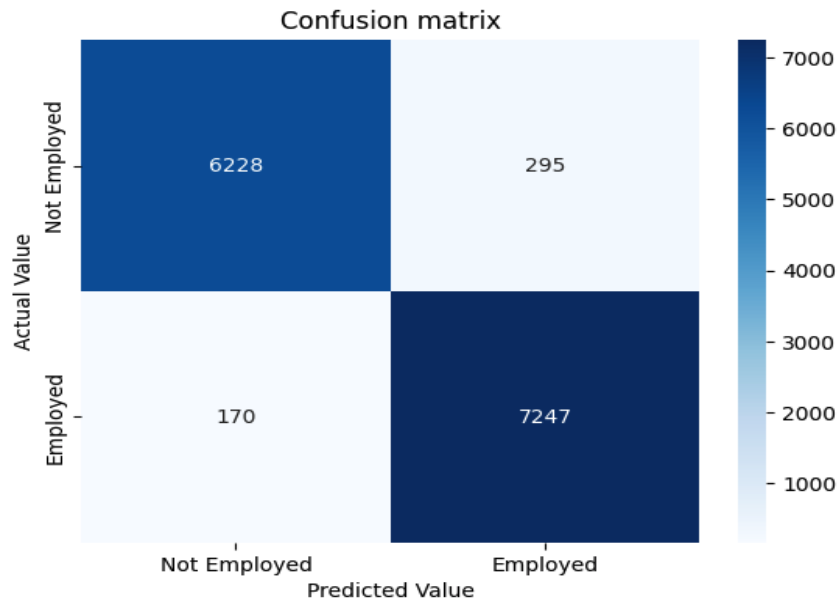
Accuracy: 96.66%

Precision: 96.09%

Recall: 97.71%

## Confusion Matrix:

The confusion matrix showed the model's high accuracy, with 6,228 true negatives and 7,247 true positives, while only misclassifying 295 as 'Employed' and 170 as 'Not Employed'.



TN: 6,228

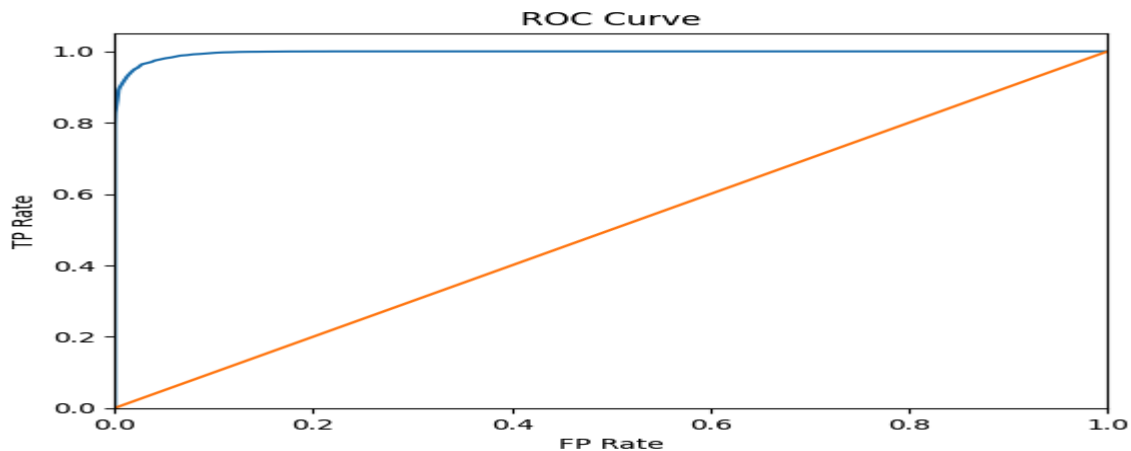
TP: 7,247

FP: 295

FN: 170

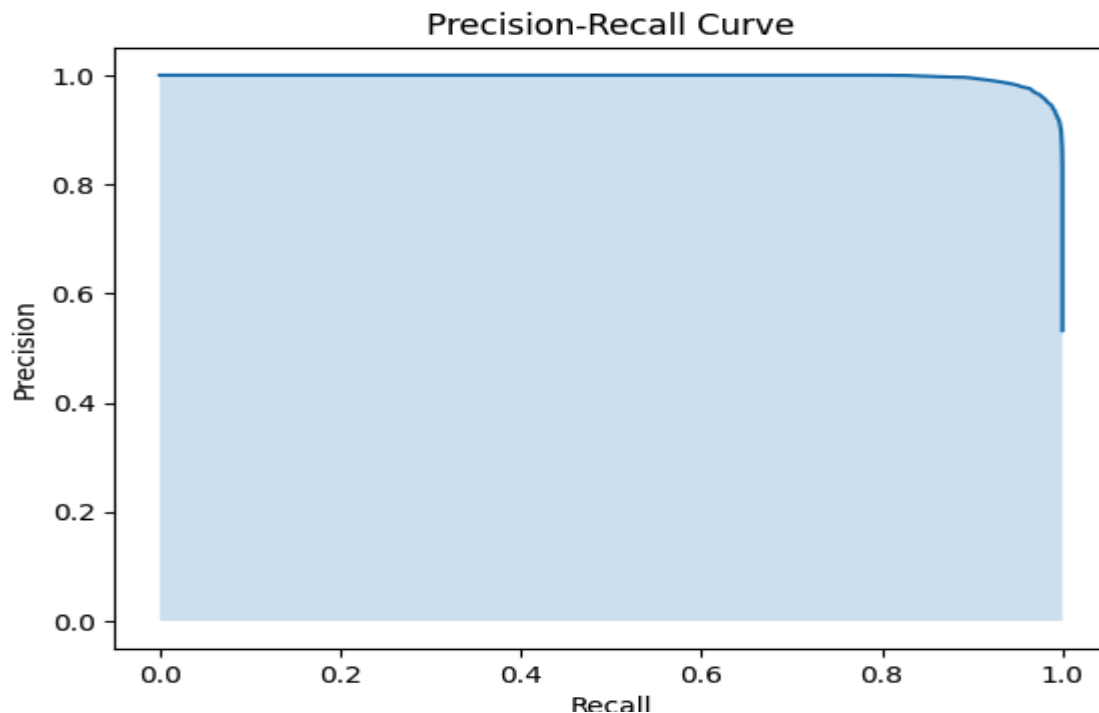
## ROC Curve:

The ROC curve, with an area of 0.997, shows the model's ability to identify between the 'Employed' and 'Not Employed' classes.



### **Precision-Recall Curve:**

The area under the Precision-Recall curve is 0.997, indicating the model's precision in predicting the 'Employed' class and its high good at handling class imbalance.



### **XGBoost:**

This algorithm uses gradient boosting for classification and it is very efficient in handling data with a large number of features with different data types.

#### **Application to Employment Status Prediction:**

XGBoost can handle a mix of categorical and numerical features which are present in our dataset such as educational background, work experience, skills, etc. It provides insights into which features are most important in predicting employability.

#### **Expectations from XGBoost:**

This model can help us understand what factors most significantly impact a candidate's suitability for a role. It includes a regularization parameters concept which can help in preventing overfitting and also ensure that the model generalizes well to new data samples.

#### **Application to Our Problem Statement:**

Here we used the scikit-learn library to train and test the model. We trained the model on a training set and used XGBClassifier to classify the test data samples. We also derived the probabilities of each class to evaluate the model's overall confidence towards its predictions.

### What we learned from the process:

The model has the perfect prediction on test data with a 100% accuracy. This shows how well XGBoost was able to handle the data.

### Performance Analysis:

Key performance metrics obtained are:

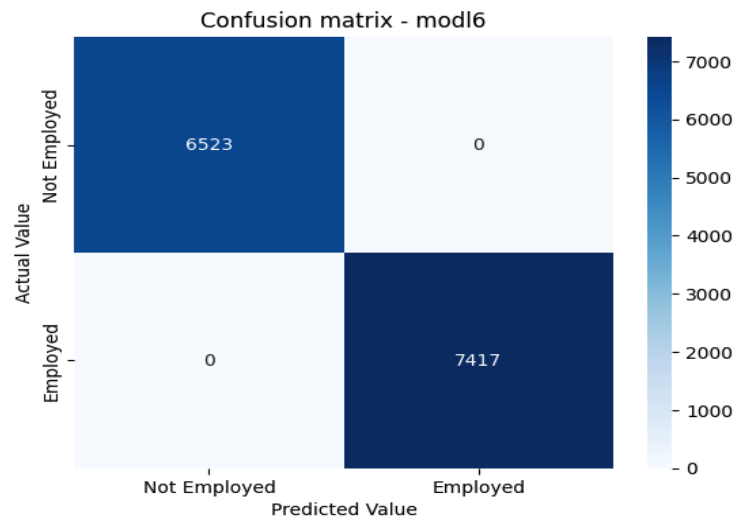
Accuracy: 100%

Precision: 100%

Recall: 100%

### Confusion Matrix:

The confusion matrix showed the model's high accuracy, with 6,523 true negatives and 7,417 true positives.



TN: 6,523

TP: 7,417

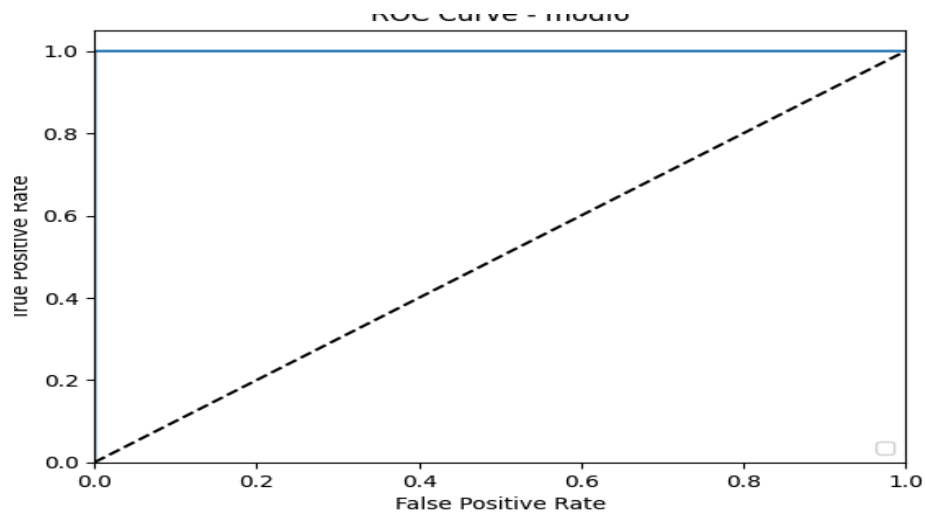
FP: 0

FN: 0

### ROC Curve:

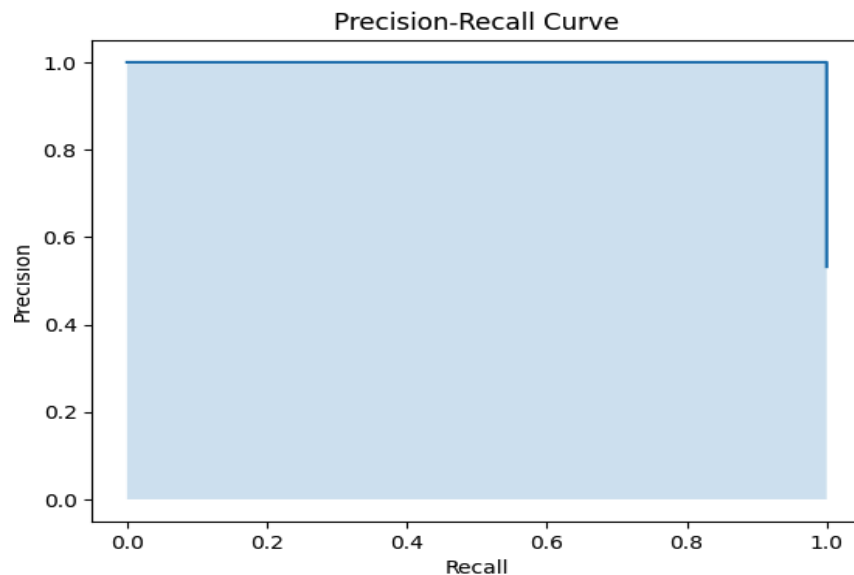
The ROC curve, with an area of 1 and a 90 degree curve, shows the model's ability to identify between the 'Employed' and 'Not Employed' classes is 100%.





### Precision-Recall Curve:

Just like, ROC curve The area under the Precision-Recall curve is 1, indicating the model's 100% precision in predicting the 'Employed'.



### Linear Regression:

Linear regression is a statistical method used for predicting continuous numerical data. The main purpose of linear regression is to find relationships between dependent variables and one or more independent variables.

#### Why Linear Regression is chosen:

We selected linear regression for predicting salary because of the linear relationship observed in the data between few features and target variable. It is also very simple and when it comes to salary prediction, it allows us to see the direct effect of applicant credentials on how much they earn.

## How we applied to our problem statement:

We started by initializing the linear regression model using scikit-learn library. We used the inbuilt fit() method to train the model and tested it on test date and got the predictions. We further did a check on the model's coefficients to understand the influence of features on predicted salary value.

## What we learned from the process:

The model provided predicted values that are both interpretable and actionable. We were able to observe the relation between features and target and some variation in how good the prediction is for different salary ranges.

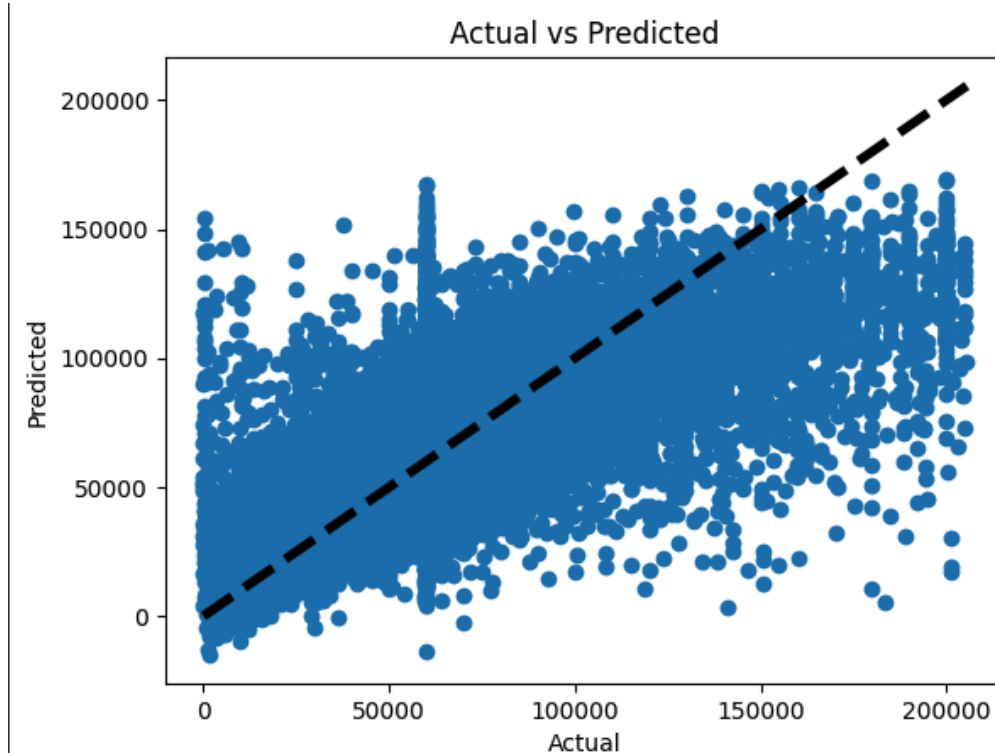
## Performance Analysis:

**MSE:** 893181021.1796179

**MAE:** 21199.35981039931

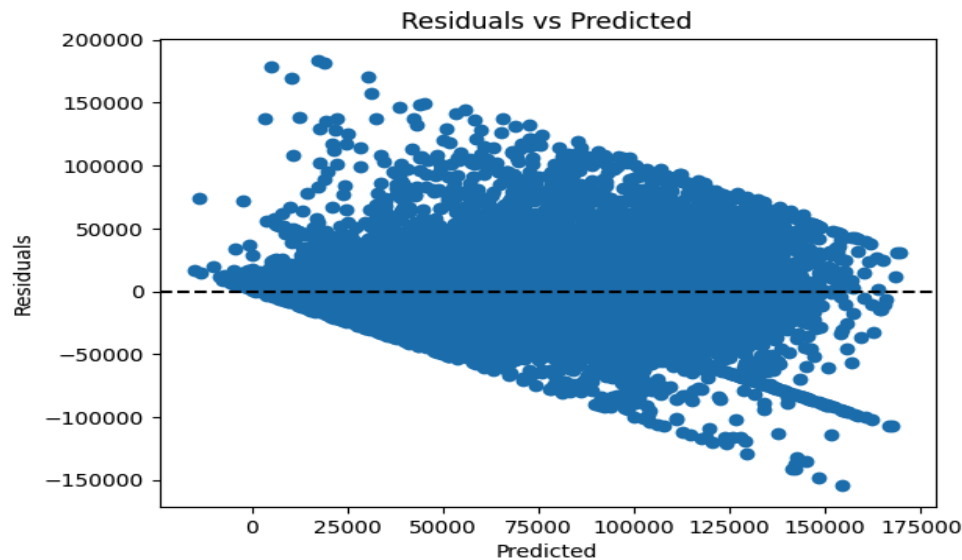
**R-squared (R2):** 0.5822607387765737

## Actual vs Predicted Graph:



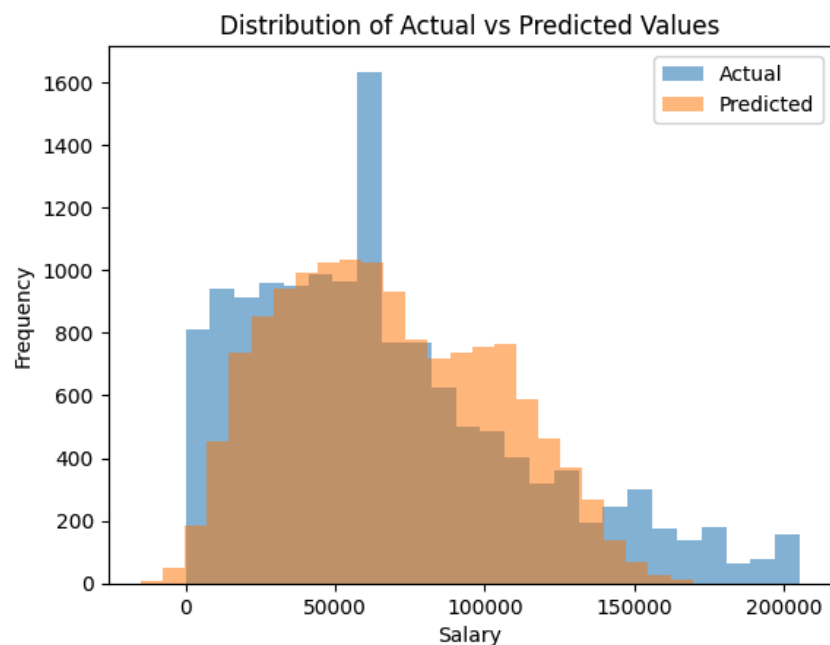
The above scatter plot shows the relationship between actual and predicted values and the dashed line show that the ideal prediction line. We see the values are close but spread across the dashed line showing variance in the predicted data from actual data. There seems to be a positive correlation but we can also say the spread is different for different salary ranges.

## Residuals vs Predicted Graph



The residuals vs predicted plot shows that difference between actual and predicted values with 0 being where predicted value equals to actual value. The spread of residuals seems to be random and variance like before is not similar across all levels of predicted values. The higher the actual salary is, the higher the difference observed.

#### Distribution of Actual vs Predicted Values Plot:



The above histogram shows the distribution frequency of actual and predicted salaries. We see a similar shape for both actual and predicted with the peak being different. The actual salary has a sharp peak but the predicted does not. This shows the model might not be checking for the variation well.

**Conclusion:**

In conclusion, the analysis of various machine learning models for employability classification indicates that the Decision Tree classifier performed well with 99% percent accuracy. and salary prediction we used linear regression.

The strength of the Decision Tree lies in its simplicity and high interpretability, which is crucial in employment decision-making contexts. Decision Tree model stands out as the most effective choice for this particular application in employability classification and linear regression model for salary prediction.

**References:**

1. [https://scikit-learn.org/stable/modules/naive\\_bayes.html](https://scikit-learn.org/stable/modules/naive_bayes.html)
2. <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>
3. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
4. [https://xgboost.readthedocs.io/en/stable/python/python\\_api.html#module-xgboost.sklearn](https://xgboost.readthedocs.io/en/stable/python/python_api.html#module-xgboost.sklearn)

# DIC PROJECT PHASE 3

**CSE 4/587B**

## **Employability Classification And Salary Estimation**

Full Name	UBIT Name	UB Number
Dharma Acha	Dharmaac	50511275
Adarsh Reddy Bandaru	adarshre	50527208
Sri Sahith Reddy Kuncharam	srisahit	50532516

## **Project Statement:**

This project mainly focuses on "Employability Classification of Job Applicants and Salary Estimation". Its main purpose is to help organizations to check if a candidate is suitable for a job opening. It also helps as a good source for the job applicants who want to know what salary to expect when applying. This solution will be mutually beneficial for the employers.

## **About the dataset:**

The dataset was taken from Kaggle and it consists of various columns, they are age, education level (EdLevel), status of the employment, gender, mental health, branch (MainBranch), years of coding experience (YearsCode and YearsCodePro), country of residence, previous salary data, work history with a particular technologies ("Have worked with"), computer skills, and status of present employment.

<https://www.kaggle.com/datasets/thedevastator/jobs-dataset-from-glassdoor>

## **Source of the data:**

By considering a period of time, this dataset has been collected from various origins, like job websites, and online applications. Data collection was conducted using standardized methods to maintain balance across various data points. This dataset has a wide variety of industries, job roles, and educational qualifications from different countries.

## **Model Selection Criteria:**

The analysis of various machine learning models for employability classification and Salary estimation indicates that the Decision Tree classifier performed well with 99% percent accuracy in predicting employability classification and for salary prediction we used linear regression model

## **Interactive Machine Learning Application( Instructions to run the application):**

Streamlit is a very common and popular Python library that simplifies the process of creating the web applications for machine learning models. Basically it is more user friendly, allowing developers to create interactable and customizable web interfaces.

So now we are going to explain some basic steps on how to use Streamlit.

First Step is to install Streamlit this can be done using the following command:

**pip install streamlit**

Next Step is to write a python script using a streamlit library to achieve the interactive UI and also we need pickles files of the model along with processed datasets which are used for UI creation.

## **Running the streamlit application**

To run the streamlit application we use the following command:

**‘streamlit run filename.py’**

The file name in our case will be “mainmodel2.py”

By using the following command we will run the application, by mentioning the filename of the streamlit python script we wrote.

Next step is to go to web browser and type

‘ <http://localhost:8501/> ’

You can also interact with the application on the following link as it is deployed:

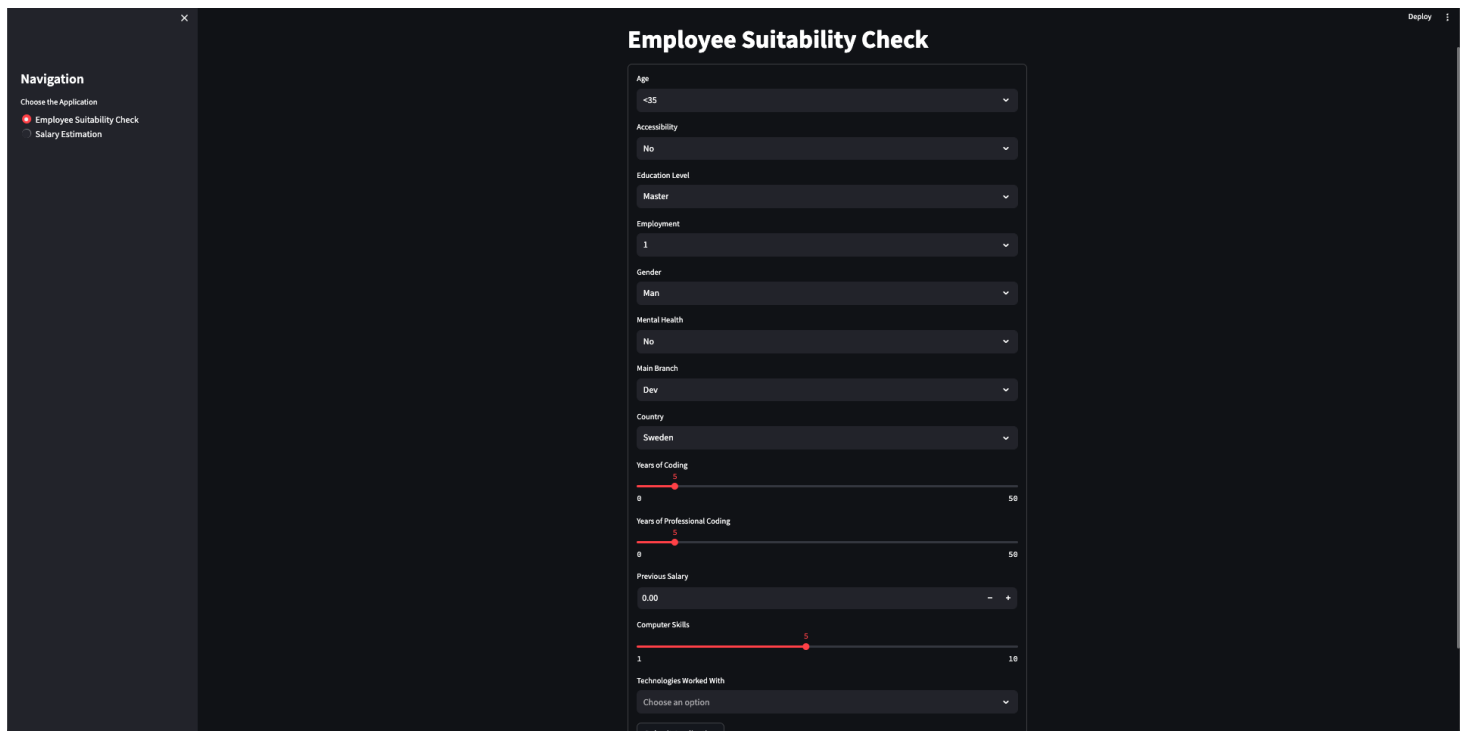
‘ <https://dicproject-3.streamlit.app/> ’

Below screenshot is of the website we created using streamlit python script.

Our website aligns according to the needs of our problem statement, which are Employability Classification of Job applications and also Salary Prediction of Job Applications.

**In Employment Classification the user should provide the following inputs**

Age  
Accessibility  
Education Level  
Gender  
Employment  
Mental Health  
Main Branch  
Country  
Years of Coding  
Years of Professional Coding  
Previous Salary  
Computer Skills  
Technologies Worked With



The screenshot displays a web application titled "Employee Suitability Check". On the left, a dark sidebar contains a "Navigation" menu with two options: "Employee Suitability Check" (selected with a red dot) and "Salary Estimation". The main content area features a series of input fields for user data: "Age" (dropdown with "<35" selected), "Accessibility" (dropdown with "No" selected), "Education Level" (dropdown with "Master" selected), "Employment" (dropdown with "1" selected), "Gender" (dropdown with "Man" selected), "Mental Health" (dropdown with "No" selected), "Main Branch" (dropdown with "Dev" selected), "Country" (dropdown with "Sweden" selected), "Years of Coding" (slider from 0 to 58 with a value of 5), "Years of Professional Coding" (slider from 0 to 58 with a value of 5), "Previous Salary" (text input with "0.00" and minus/plus buttons), "Computer Skills" (slider from 1 to 18 with a value of 5), and "Technologies Worked With" (dropdown with "Choose an option" selected). A "Submit Application" button is located at the bottom of the form. A "Deploy" button is visible in the top right corner of the application interface.

When we press the submit application button, we get the result whether the job applicant is suitable, based on skills they have. We have designed radio buttons to switch easily between two features that we have on the website, one is employee suitability check and the other one is salary estimation. Below is the screenshot of how the interface looks for salary estimation.

The screenshot shows a web application titled "Salary Estimation". On the left is a dark sidebar with a "Navigation" section containing two radio buttons: "Employee Suitability Check" (unselected) and "Salary Estimation" (selected). The main content area has a light background and contains the following form elements:

- Education Level:** A dropdown menu with "Master" selected.
- Gender:** A dropdown menu with "Man" selected. Below it are buttons for "Man", "NonBinary", "Woman", and "Other".
- Years of Professional Coding:** A horizontal slider ranging from 0 to 50, with a red dot at 9.
- Computer Skills:** A horizontal slider ranging from 1 to 10, with a red dot at 5.
- Technologies Worked With:** A collection of red tags with white text and close buttons: "python", "postgresql", "mysql", "typescript", "git", "aws", "sql", "node.js", "java", "bash/shell", "c#", "microsoft sql ser...", "sqlite", and "react.js".
- Employed:** A dropdown menu with "1" selected.
- Submit Salary Estimation:** A button at the bottom of the form.

## Inputs user should provide for salary estimation

Education Level  
Gender  
Country  
Years of Coding  
Years of Professional Coding  
Computer Skills  
Technologies Worked With  
Employed

## Predicting the Employability Classification using Decision Tree model :

```
import streamlit as st
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```



```
import time
```

We have imported all the required libraries for the application to run such as streamlit, pandas, numpy, pickle, matplotlib.pyplot seaborn, plotly.express, time.

```
with open('decision_tree_model.pkl', 'rb') as file:  
    decision_tree_model = pickle.load(file)
```

Using the above line we are loading the pickle file of the decision tree model, which has been created in phase 2 of the project, where we decided this would be the best model to predict the employability classification.

```
employability_data = pd.read_csv('model_data.csv')  
full_data = pd.read_csv('stackoverflow_full.csv')
```

We have loaded the above preprocessed data which we are using for showing unique values under the dropdowns in the UI. This will enable the users to select the columns like skills, gender, and others to avoid unnecessary errors.

```
skills_list_employability = [col for col in employability_data.columns if not col.startswith(('Country_', 'Age', 'Accessibility',  
'EdLevel', 'Employment', 'Gender', 'MentalHealth', 'MainBranch', 'YearsCode', 'YearsCodePro', 'ComputerSkills',  
'Employed', 'PreviousSalary'))]  
min_max_values_employability = {  
    'YearsCode': (full_data['YearsCode'].min(), full_data['YearsCode'].max()),  
    'YearsCodePro': (full_data['YearsCodePro'].min(), full_data['YearsCodePro'].max()),  
    'PreviousSalary': (full_data['PreviousSalary'].min(), full_data['PreviousSalary'].max()),  
    'ComputerSkills': (full_data['ComputerSkills'].min(), full_data['ComputerSkills'].max())  
}
```

Here we are gathering min and max values of all the numerical columns to be able to process the inputs that the user gives and apply normalization to it.

```
# Process user inputs checking employee suitability  
def process_input_employability(user_data):  
    input_df = pd.DataFrame([user_data])  
    for skill in skills_list_employability:  
        input_df[skill] = 1 if skill in user_data['Skills'] else 0  
  
    #normalization  
    for col, (min_val, max_val) in min_max_values_employability.items():  
        input_df[col] = (input_df[col] - min_val) / (max_val - min_val)  
  
    #one-hot encoding
```

```

categorical_cols = ['Age', 'Accessibility', 'EdLevel', 'Employment', 'Gender', 'MentalHealth', 'MainBranch', 'Country']
input_df = pd.get_dummies(input_df, columns=categorical_cols, prefix=categorical_cols)

final_df = pd.DataFrame(columns=[col for col in employability_data.columns if col != 'Employed'])
for col in final_df.columns:
    final_df[col] = input_df[col] if col in input_df else 0
return final_df

```

To be able to match the user input to what the model is expecting, we are going to perform one-hot encoding to the input. Lastly, we are going to match the columns from input df and preprocessed data and remove unnecessary columns.

```

# Employability Suitability Check
if page == "Employee Suitability Check":
    st.title('Employee Suitability Check')

    with st.form("Employee Suitability Check Form"):
        age = st.selectbox('Age', extract_unique_values(full_data, 'Age'))
        accessibility = st.selectbox('Accessibility', extract_unique_values(full_data, 'Accessibility'))
        ed_level = st.selectbox('Education Level', extract_unique_values(full_data, 'EdLevel'))
        employment = st.selectbox('Employment', extract_unique_values(full_data, 'Employment'))
        gender = st.selectbox('Gender', extract_unique_values(full_data, 'Gender'))
        mental_health = st.selectbox('Mental Health', extract_unique_values(full_data, 'MentalHealth'))
        main_branch = st.selectbox('Main Branch', extract_unique_values(full_data, 'MainBranch'))
        country = st.selectbox('Country', extract_unique_values(full_data, 'Country'))
        years_code = st.slider('Years of Coding', 0, 50, 5)
        years_code_pro = st.slider('Years of Professional Coding', 0, 50, 5)
        previous_salary = st.number_input('Previous Salary')
        computer_skills = st.slider('Computer Skills', 1, 10, 5)
        skills = st.multiselect('Technologies Worked With', skills_list_employability)

        submitted = st.form_submit_button("Submit Application")

```

The above code is to take inputs from the user using form in streamlit which is similar to html form.

**Screenshot of the form:**

Then Upon submitting using the form submit button which is named as “Submit Application”. We will be sending the data given by the user to the model to predict.

```
if submitted:
    st.session_state['employability_submitted'] = True
    progress_bar = st.progress(0)
    for i in range(100):
        time.sleep(0.01)
        progress_bar.progress(i + 1)

    user_data = {
        'Age': age,
        'Accessibility': accessibility,
        'EdLevel': ed_level,
        'Employment': employment,
        'Gender': gender,
        'MentalHealth': mental_health,
        'MainBranch': main_branch,
        'Country': country,
        'YearsCode': years_code,
        'YearsCodePro': years_code_pro,
        'PreviousSalary': previous_salary,
        'ComputerSkills': computer_skills,
        'Skills': skills
    }
```

The above map we created, which is **user\_data**, is used to send the data to a model pkl file after passing it through process\_input\_employability which can be seen below.

```
processed_input = process_input_employability(user_data)
employability = decision_tree_model.predict(processed_input)[0]
```

This line of code above, is where we are sending the data to the model to predict.

```
st.write('Employee Suitability Check:', 'Candidate is Suitable' if employability == 1 else 'Candidate is Not Suitable')
```

This line of code actually prints if the job applicant is suitable or not suitable.

**Below are the outputs:**

Employee is Suitable

Technologies Worked With

angular.js ×

aws ×

bash/shell ×

typescript ×

javascript ×

react.js ×

vue.js ×

node.js ×

postgresql ×

git ×

express ×

html/css ×

×

▼

Submit Application

Employee Suitability Check: Candidate is Suitable

Employee is Not Suitable

Technologies Worked With

node.js ×

postgresql ×

git ×

express ×

html/css ×

×

▼

Submit Application

Employee Suitability Check: Candidate is Not Suitable

```
st.subheader('Years of Coding by Age and Gender')
fig1, ax1 = plt.subplots()
sns.violinplot(x='Age', y='YearsCode', hue='Gender', data=full_data, ax=ax1)
ax1.set_title('Years of Coding by Age and Gender')
ax1.set_xlabel('Age')
ax1.set_ylabel('Years of Coding')
ax1.tick_params(axis='x', rotation=45)
```

```

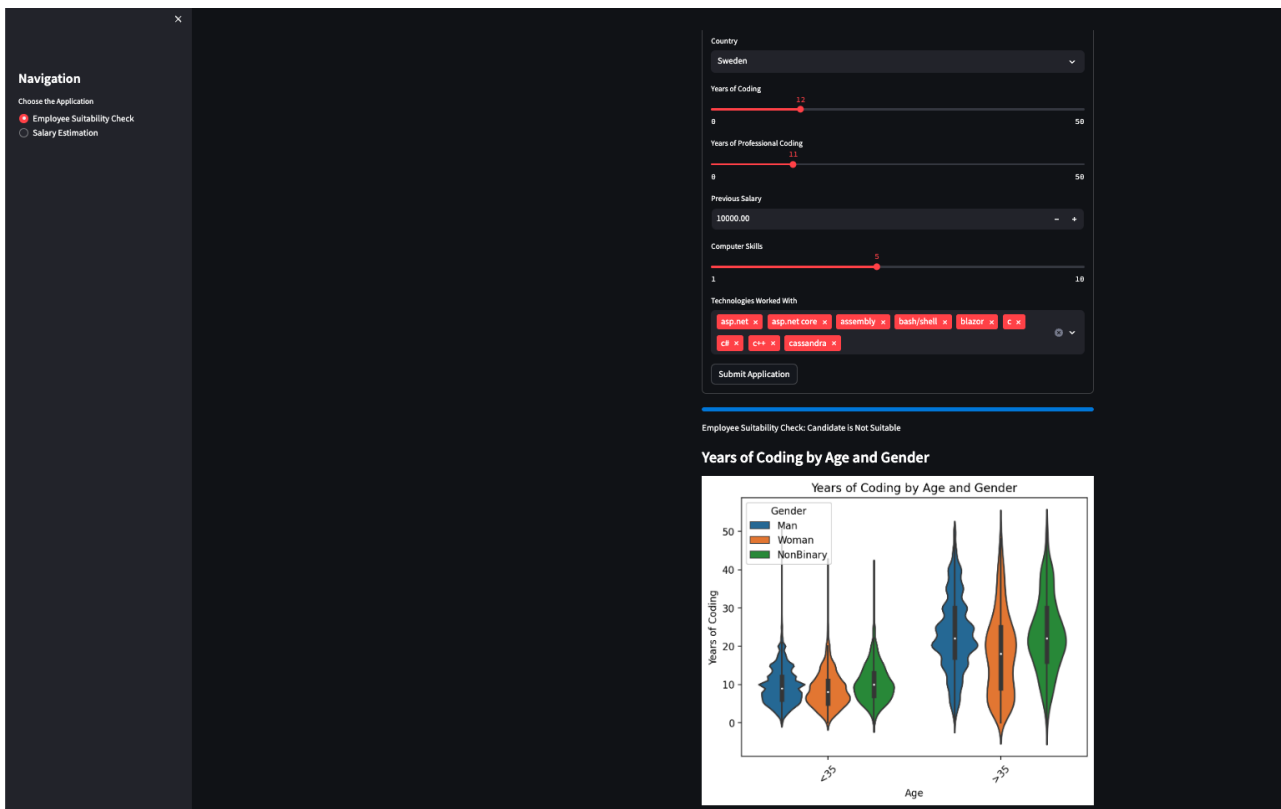
st.pyplot(fig1)

# EDA: Years of Coding Experience by Education Level
st.subheader('Years of Coding Experience by Education Level')
fig2, ax2 = plt.subplots()
sns.boxplot(x="EdLevel", y="YearsCode", data=full_data, ax=ax2)
ax2.set_title('Years of Coding Experience by Education Level')
ax2.set_xlabel('Education Level')
ax2.set_ylabel('Years of Coding')
ax2.tick_params(axis='x', rotation=45)
st.pyplot(fig2)

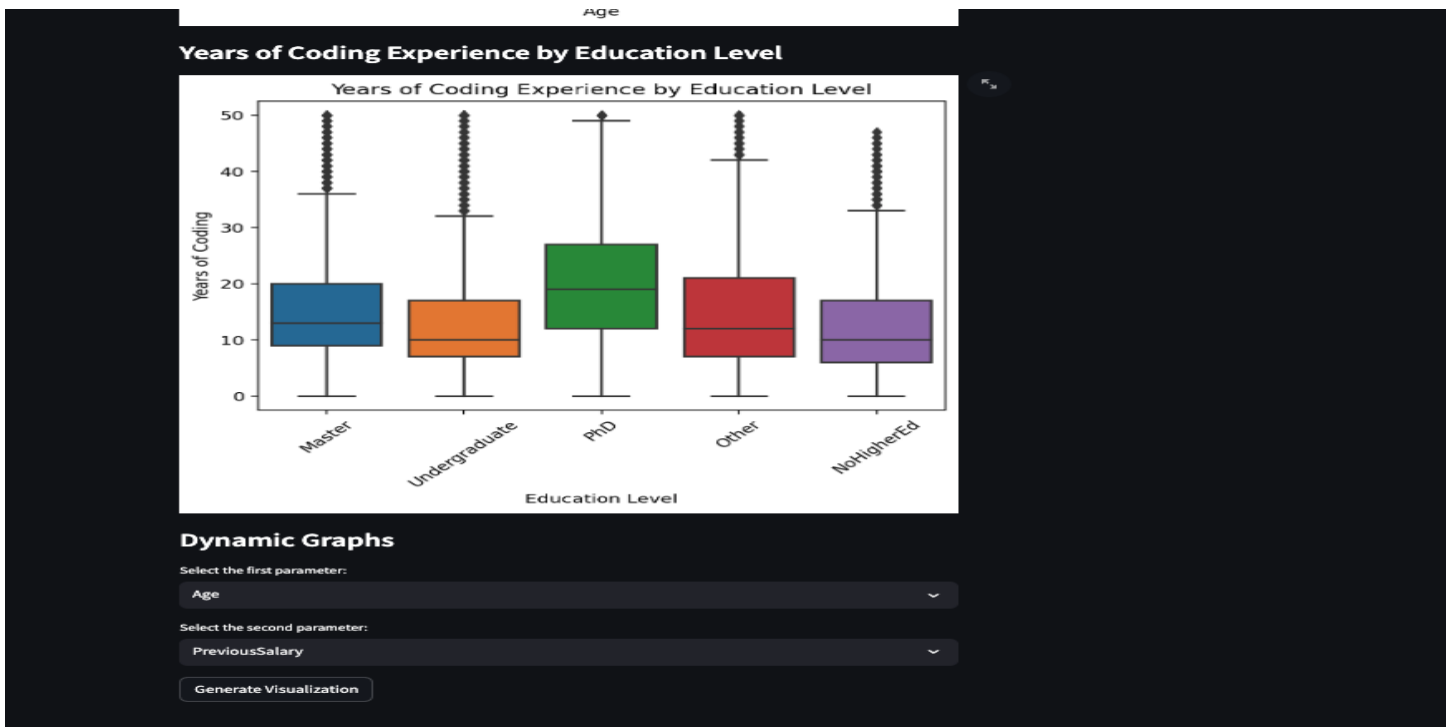
if st.session_state.get('employability_submitted', False):
    st.subheader('Dynamic Graphs')

```

Above code will show the visualization of the years of coding by age and gender, and also years of coding experience by education level.



In the years of coding by Age and Gender we have two categories in age, first category is age less than age 35 and second category is where age greater than 35.



Second Visualization shows the distribution among the education level v/s years of coding experience.

```
# Dynamic graphs
option_1 = st.selectbox('Select the first parameter:', ['Age', 'Gender', 'EdLevel'], key='dynamic_option_1')
option_2 = st.selectbox('Select the second parameter:', ['PreviousSalary', 'YearsCode', 'YearsCodePro', 'ComputerSkills'],
key='dynamic_option_2')

if st.button('Generate Visualization'):
    if option_1 in full_data and option_2 in full_data:
        plt.figure(figsize=(10, 6))
        sns.violinplot(data=full_data, x=option_1, y=option_2)
        plt.title(f'{option_1} vs {option_2} (Violin Plot)')
        plt.xlabel(option_1)
        plt.ylabel(option_2)
        st.pyplot(plt)
    else:
        st.error("Invalid column selection. Please select different parameters.")
```

Finally, this lines of code will show us the dynamic graphs where we have drop down menu to choose over different combinations

We have implemented this feature in such a way that we can choose over 7 parameters, 3 categorical and 4 numerical.

Where we can choose two of them to get the violin plot to see the density at specific groups in the data. i.e. age, gender, edlevel.

### Dynamic Graphs

Select the first parameter:

Age

Age

Gender

EdLevel

v/s  
previous salary, yearscode, yearscodepro, computerskills.

Select the second parameter:

PreviousSalary

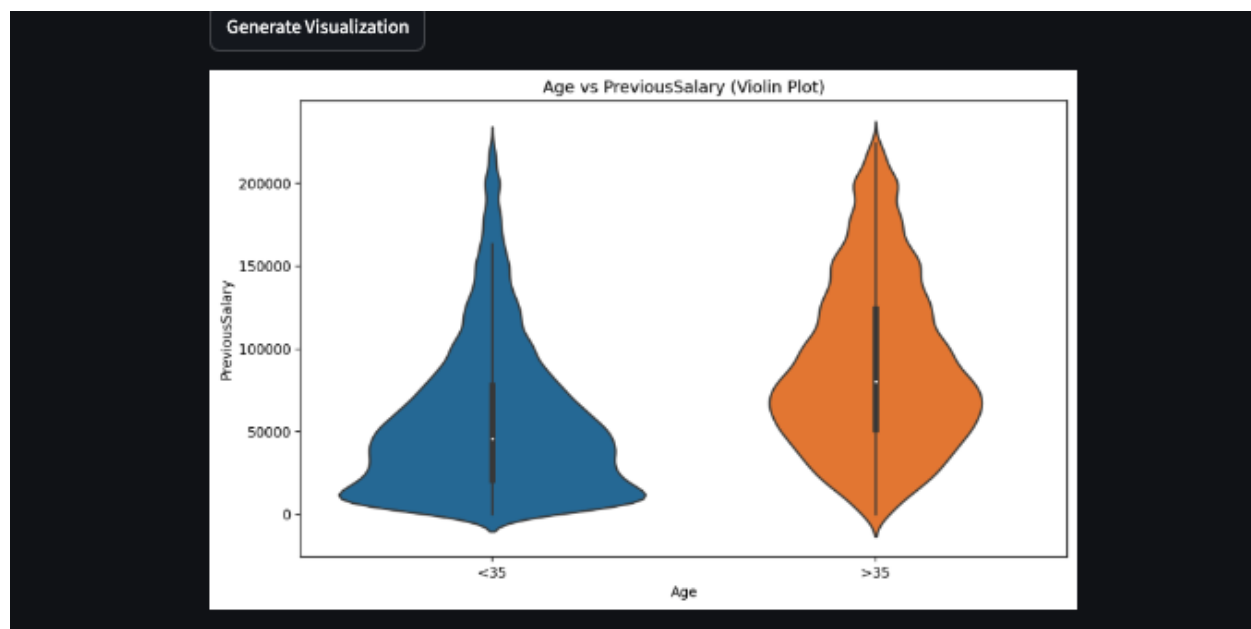
PreviousSalary

YearsCode

YearsCodePro

ComputerSkills

### Sample of the dynamic graphs



This graph can help us understand the general trends and comparison.

## Predicting the Salary using Linear Regression model :

Loading the necessary python libraries:

```
import streamlit as st
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import time
import streamlit as st
```

The first section of code begins with loading of pre-trained model Linear regression which we used for salary prediction and loads processed salary data. These steps can be done by following below steps:

The initial step is to open the pickle file related to Linear regression with the help of **with open (User/file\_path/regression.pkl)**. The preprocessed data set of Salary is loaded using **pd.read\_csv(User/file\_paths/processed\_salary\_data.csv)**.

The salary prediction of an employee begins with collecting user\_data and the code follows below screenshot.

```
def process_input_salary(user_data):
    input_df = pd.DataFrame([user_data])

    for skill in skills_list_salary:
        input_df[skill] = 1 if skill in user_data['Skills'] else 0
    # normalization
    for col, (min_val, max_val) in min_max_values_salary.items():
        input_df[col] = (input_df[col] - min_val) / (max_val - min_val)

    # One-hot encoding
    categorical_cols = ['EdLevel', 'Gender', 'Country']
    input_df = pd.get_dummies(input_df, columns=categorical_cols, prefix=categorical_cols)

    final_df = pd.DataFrame(columns=[col for col in salary_data.columns if col != 'PreviousSalary'])
    for col in final_df.columns:
        final_df[col] = input_df[col] if col in input_df else 0
    return final_df
```

1. The function process\_input\_salary takes user\_input like education level, gender, country, years of coding experience, computer skills and many other features.



2. After utilizing the user\_input parameters it processes it into Dataframe suitable for making predictions with the Salary prediction model which uses Linear regression model.
3. And this also involves handling skills, Scaling numerical columns and performing one-hot encoding for categorical data.

```
# Salary Estimation Page
elif page == "Salary Estimation":
    st.title('Salary Estimation')

    with st.form("salary_estimation_form"):
        ed_level = st.selectbox('Education Level', ['Master', 'NoHigherEd', 'Other', 'PhD', 'Undergraduate'])
        gender = st.selectbox('Gender', ['Man', 'NonBinary', 'Woman', 'Other'])
        country = st.selectbox('Country', extract_unique_values(full_data, 'Country'))
        years_code = st.slider('Years of Coding', 0, 50, 5)
        years_code_pro = st.slider('Years of Professional Coding', 0, 50, 5)
        computer_skills = st.slider('Computer Skills', 1, 10, 5)
        skills = st.multiselect('Technologies Worked With', skills_list_salary)
        employed = st.selectbox('Employed', [0, 1])

    submitted = st.form_submit_button("Submit Salary Estimation")
```

4. The above code collects the user inputs from the Streamlit interface for salary prediction. The users(Employers) now can select education level, gender, country, years of coding experience, years of professional coding, computer skills, and the technologies employees have worked before. And the submit salary estimation button triggers the prediction process.

```
user_data = {
    'YearsCode': years_code,
    'YearsCodePro': years_code_pro,
    'ComputerSkills': computer_skills,
    'EdLevel': f'EdLevel_{ed_level}',
    'Gender': f'Gender_{gender}',
    'Country': country,
    'Skills': skills,
    'Employed': employed
}
```

5. The Streamlit web application which uses Linear regression classifier to predict the salary of the employee can be seen below. This interface is very interactable as it is seen the employer can select the education level, gender, Country by using a drop down. We implemented this by using Streamlit web interface templates (selectbox). For better understanding see the below code.

```
ed_level = st.selectbox('Education Level', ['Master', 'NoHigherEd', 'Other', 'PhD', 'Undergraduate'])
gender = st.selectbox('Gender', ['Man', 'NonBinary', 'Woman', 'Other'])
```

```
country = st.selectbox('Country', extract_unique_values(full_data, 'Country'))
```

6. The actual Streamlit web page looks as below for Salary Estimation.

7. And for the Years of coding , years of professional coding and computer skills we used “slider”. See below how we implemented it in the code.

```
years_code = st.slider('Years of Coding', 0, 50, 5)
years_code_pro = st.slider('Years of Professional Coding', 0, 50, 5)
computer_skills = st.slider('Computer Skills', 1, 10, 5)
```

8. And for technologies working with fields we used “multiselect”. See below for better understanding

```
skills = st.multiselect('Technologies Worked With', skills_list_salary)
```

9. The below screenshot represents how we select multiple technologies.

10. As we hit the Submit Salary prediction button the model predicts the salary of the employee with scrolling blue animation. See the below screenshot to know the value of the predicted salary of the employee.

Gender

Man

Country

Sweden

Years of Coding

9

0 50

Years of Professional Coding

9

0 50

Computer Skills

5

1 10

Technologies Worked With

python x postgresql x mysql x typescript x git x aws x sql x

node.js x java x bash/shell x c# x microsoft sql ser... x sqlite x

react.js x

Employed

1

Submit Salary Estimation

Salary Prediction: 47825.31707119108

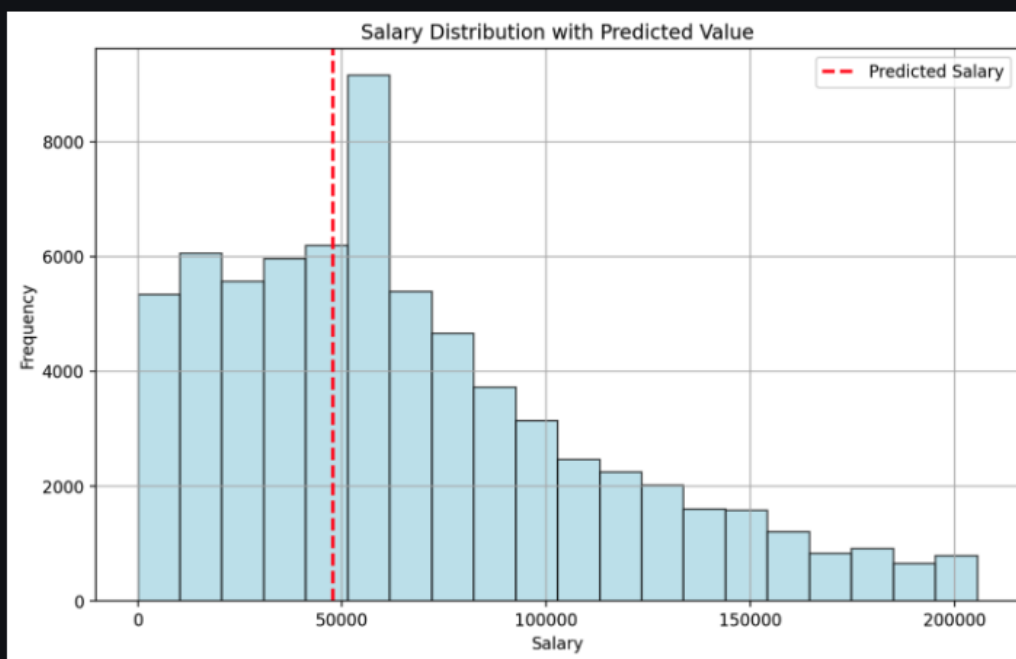
## User interface Visualizations:

### 1. The Histogram Plot showing the predicted salary value in the range of Salaries Distribution.

The plot below represents the salary distribution of employees. The blue bars represents the range of salaries and the predicted salary of the employee is marked with orange dashed line which falls slightly below the 50000 which is equal to 47825.317.

Overall the plot suggests that the majority of employees earn between \$ 50000 to \$ 100000. The predicted values

are very close to the actual values, which suggest that the model is able to accurately predict employee salaries.



## 2. Pair plots of Selected Columns:

The pair plot is helpful for selecting any number of columns (features) .With the help of pairplot we are showing the relation between the selected pair of columns.

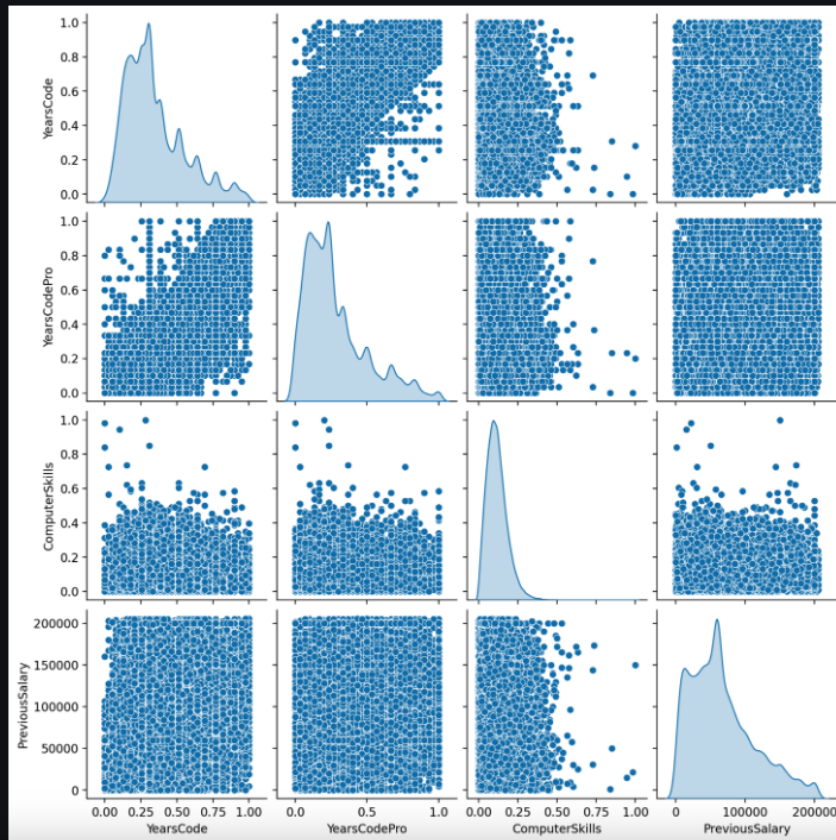
In this User interface, we have chosen YearsCode, YearCodePro, ComputerSkills, Previous Salary has pair plots. The graph is plotted among them by selecting any two pairs from selected columns.

The scatter plot shows that there is positive correlation between the two variables YearsCode and Computer skills, that means increase in YearsCode leads to increase in computer skills this represents that more experienced coders have better computer skills

Select Columns for Pair Plot

YearsCode × YearsCodePro × ComputerSkills × PreviousSalary ×

### Pair Plot of Selected Columns



### What users learn from the product:

1. For the Employability classification, the users can learn whether an employee is suitable for a job opening based on their input information. So our model predicts employability, providing valuable insights into the suitability of being selected.
2. Job applicants can know their employability and understand the potential areas for improvement, and also in the perspective of employers there is a chance to quickly evaluate the suitability of applicants for their organizations which helps them to minimize the selection process of the candidates.
3. Users can estimate the salary based on their input parameters, which helps them to set realistic expectations
4. Applicants can learn how to negotiate salaries effectively with their employers by using this product.
5. Users can visually explore dynamic graphs to understand relationships between different parameters.
6. The visual exploration allows users to identify trends and patterns and correlation in the data, making a deeper understanding of the factors influencing employability and salary predictions.

### Ideas for Extension and Exploration:

1. **Feature Enhancement:**
  - a. **Recommendation Engine:** Implement a recommendation engine that gives additional skills or qualification based on user's profile which helps in enhancing their employability
  - b. **Career Path Guidance:** Extending the application to provide the career path guidance and also

suggesting the roles or organizations aligned with the user's skills and experience.

- c. **Real-Time Job market Insights:** Provide real-time insights into job market trends and also showing specific skills and average salaries demand in various companies.

## 2. User interaction and Personalization:

- a. **Interactive Dashboards:** Develop interactive dashboards that allow users to customize the visualizations and explore particular aspects of the information related to their interests.
- b. **User Profiles:** Allow users to create profiles to track their employment history, skills and salary progression over time which may act as career management tool.

## Application Link:

<https://dicproject-3.streamlit.app/>

## References:

<https://streamlit.io/>

<https://docs.streamlit.io/library/get-started/installation>

<https://streamlit.io/components?category=text>

<https://okld-gallery.streamlit.app/?p=ace-editor>