

# **SOFTWARE SALARY PREDICTION**

## **Final Project Report**

### **1. Introduction**

#### 1.1. Project Overview

#### 1.2. Objectives

### **2. Project Initialization and Planning Phase**

#### 2.1. Define Problem Statement

#### 2.2. Project Proposal (Proposed Solution)

#### 2.3. Initial Project Planning

### **3. Data Collection and Preprocessing Phase**

#### 3.1. Data Collection Plan and Raw Data Sources Identified

#### 3.2. Data Quality Report

#### 3.3. Data Exploration and Preprocessing

### **4. Model Development Phase**

#### 4.1. Feature Selection Report

#### 4.2. Model Selection Report

## **5.Model Training and Evaluation Phase**

### 5.1. Training Results

### 5.2. Model Evaluation Metrics

### 5.3. Final Model Selection Justification

## **6.Results**

### 6.1. Output Screenshots

## **7.Advantages & Disadvantages**

## **8.Conclusion**

## **9.Future Scope**

## **10.Appendix**

### 10.1. Source Code

# 1.INTRODUCTION

## 1.1.OVERVIEW

Software salary prediction uses data-driven methods to estimate the earnings of software professionals, factoring in experience, location, education, job role, and skills. In the rapidly evolving tech industry, understanding salary trends is vital for employers to create competitive compensation packages and for employees to assess their market value. Leveraging machine learning and data analytics, predictive models analyze historical salary data to uncover trends and patterns. These models incorporate diverse features, from demographics to market conditions and technological advancements, enhancing the accuracy of predictions. Accurate salary predictions enable informed decision-making, aiding companies in competitive hiring and helping professionals make strategic career choices. This overview explores the methodologies, data sources, and challenges of software salary prediction, emphasizing the role of advanced analytics in navigating the complexities of the tech job market and anticipating salary trends.

## 1.2.PURPOSE

The purpose of predicting software salaries across different sectors is multi-faceted, addressing the needs of various stakeholders in the tech industry.

For Job Seekers:

Understanding salary expectations in different sectors enables job seekers to make informed career choices. By comparing potential earnings in sectors such as finance, healthcare, technology, and education, professionals can identify where their skills and experience are most valued. This information can also guide their efforts in acquiring additional qualifications or specializations that are in high demand in more lucrative sectors.

For Employers:

Employers can utilize salary prediction models to design competitive compensation packages that attract and retain top talent. By benchmarking their salary offerings against industry standards and understanding sector-specific salary trends, companies can ensure they remain competitive in the labor market. This is particularly important in sectors where the demand for skilled software developers is high, and the supply is limited.

## For Educators and Career Counselors:

Educational institutions and career counselors can use these insights to guide students and professionals towards sectors with higher earning potentials. By aligning educational programs and career advice with market demands, they can better prepare individuals for successful careers in the tech industry.

## For Industry Analysts and Policymakers:

Accurate salary predictions provide valuable data for industry analysts and policymakers. Understanding salary trends across sectors can help in workforce planning, economic forecasting, and developing policies that address skill gaps and promote equitable growth within the tech industry.

In essence, predicting software salaries in different sectors facilitates informed decision-making, promotes career growth, and helps maintain a competitive and balanced job market.

## 2.LITERATURE SURVEY

### 2.1 EXISTING PROBLEM

The existing problem of salary prediction in the software industry is a multifaceted challenge that spans economic, technological, and social domains. Inaccurate salary predictions can lead to significant disparities in compensation, affecting employee satisfaction and retention.

Moreover, discrepancies in salary prediction can create a lack of transparency and trust between employers and employees. It can also contribute to broader issues such as wage gaps across different demographics, including gender and ethnicity. Economically, inaccurate salary predictions can result in inefficient resource allocation, affecting both company profitability and employee livelihood. Additionally, these disparities can exacerbate socio-economic inequalities, placing vulnerable groups at a disadvantage.

Inadequate data quality, limited access to comprehensive datasets, and the variability of salary determinants further compound the issue. Addressing these problems necessitates stringent data validation processes, the adoption of advanced predictive modeling techniques, robust education on fair compensation practices, and international cooperation to standardize salary prediction methods to ensure fair and equitable compensation in the software industry.

### 2.2 PROPOSED SOLUTION

Our innovative proposed solution leverages advanced predictive modeling to anticipate and communicate software salary ranges accurately and efficiently. By forecasting salary ranges, we enable proactive dissemination of information regarding compensation trends, accompanying industry benchmarks, and recommended salary negotiations. This solution integrates technology, data analytics, and public awareness to create a comprehensive system for managing salary prediction issues.

1. **Advanced Predictive Modeling:** We employ sophisticated predictive models that take into account various factors such as industry trends, historical salary data, and real-time job market monitoring to forecast salary ranges. This ensures timely and reliable information for employers and job seekers

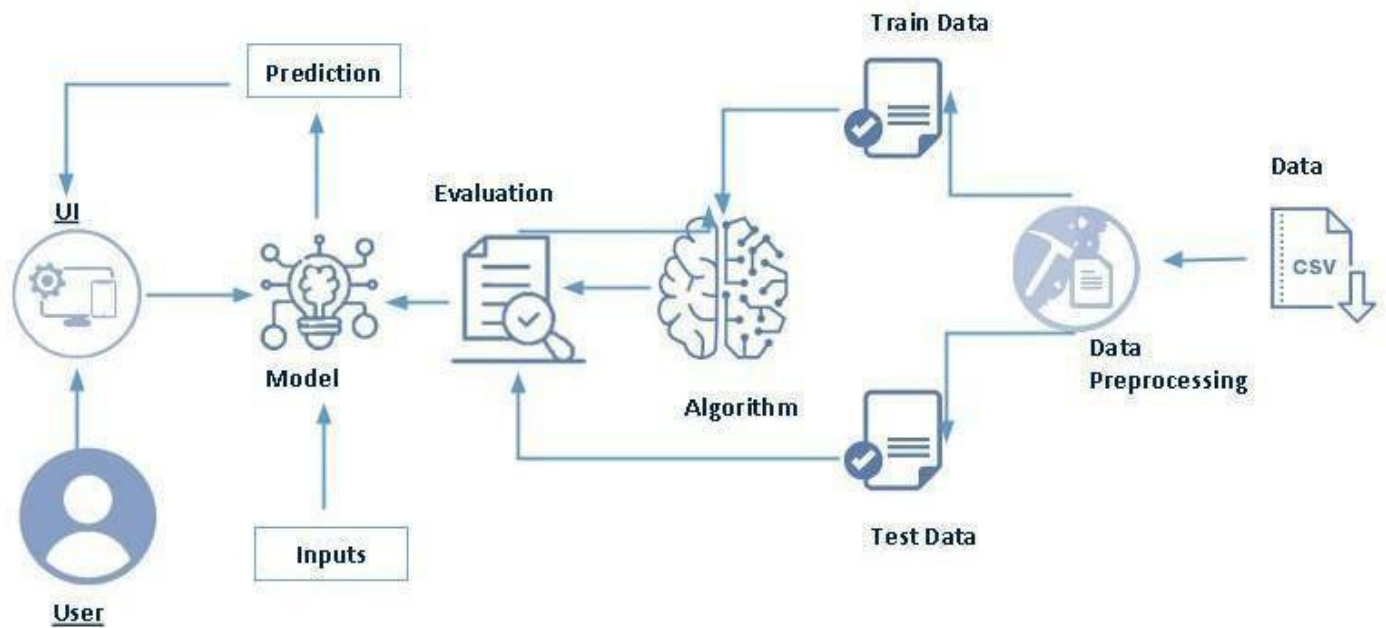
2. **Customized Compensation Information:** For each job role and experience level, our system provides a tailored list of potential salary ranges and recommended compensation packages. This empowers individuals to take proactive steps in salary negotiations based on the forecasted compensation trends.
3. **User-Friendly Interfaces:** Our solution offers user-friendly interfaces, making it easy for employers and job seekers to access and understand salary predictions and associated compensation information.
4. **Community Engagement:** We actively engage with the software industry community to raise awareness about the salary forecasting system, ensuring that employers and job seekers are well-informed and prepared to take necessary steps in salary negotiations.
5. **Educational Initiatives:** Our solution includes educational initiatives to raise awareness about the importance of fair compensation and the impact of salary transparency on employee satisfaction and retention, empowering individuals to make informed choices.

By accurately predicting salary ranges and providing associated compensation information, our solution empowers individuals and organizations to make informed decisions, ensure fair compensation, and reduce the adverse effects of salary discrepancies. It is a comprehensive approach to proactively address salary prediction issues and create equitable compensation practices for all.

By encompassing these elements, our proposed solution creates a comprehensive and adaptable framework for salary prediction and management in the software industry, with a strong emphasis on accuracy, user-friendliness, and community engagement.

### 3.THEORITICAL ANALYSIS

#### 3.1 BLOCK DIAGRAM



#### 3.2.SOFTWARE DESIGNING

The following is the Software required to complete this project:

**Google Colab:** Google Colab will serve as the development and execution environment for your predictive modeling, data preprocessing, and model training tasks. It provides a cloud-based Jupyter Notebook environment with access to Python libraries and hardware acceleration.

**Dataset (CSV File):** The dataset in CSV format is essential for training and testing your predictive model. It should include historical salary data, job titles, experience levels, location, education background, and other relevant features.

**Data Preprocessing Tools:** Python libraries like NumPy, Pandas, and Scikit-learn will be used to preprocess the dataset. This includes handling missing data, feature scaling, and data cleaning.

**Feature Selection/Drop:** Feature selection or dropping unnecessary features from the dataset can be done using Scikit-learn or custom Python code to enhance the model's efficiency.

**Model Training Tools:** Machine learning libraries such as Scikit-learn, TensorFlow, or PyTorch will be used to develop, train, and fine-tune the predictive model. Regression models can be considered for predicting salary based on the input features.

**Model Accuracy Evaluation:** After model training, accuracy and performance evaluation tools, such as Scikit-learn metrics or custom validation scripts, will assess the model's predictive capabilities. You'll measure the model's ability to predict salaries based on historical data.

**UI Based on Flask Environment:** Flask, a Python web framework, will be used to develop the user interface (UI) for the system. The Flask application will provide a user-friendly platform for users to input job-related data or view salary predictions and related information.

**Integration of Google Colab and Flask:** Google Colab will be the central hub for model development and training, while Flask will facilitate user interaction and data presentation. The dataset, along with data preprocessing, will ensure the quality of the training data, and feature selection will optimize the model. Finally, model accuracy evaluation will confirm the system's predictive capabilities, allowing users to rely on the salary predictions and associated information.



## 4.EXPERIMENTAL INVESTIGATION

In this project, we have used a Software Salary Dataset. This dataset is a CSV file consisting of labeled data and having the following columns:

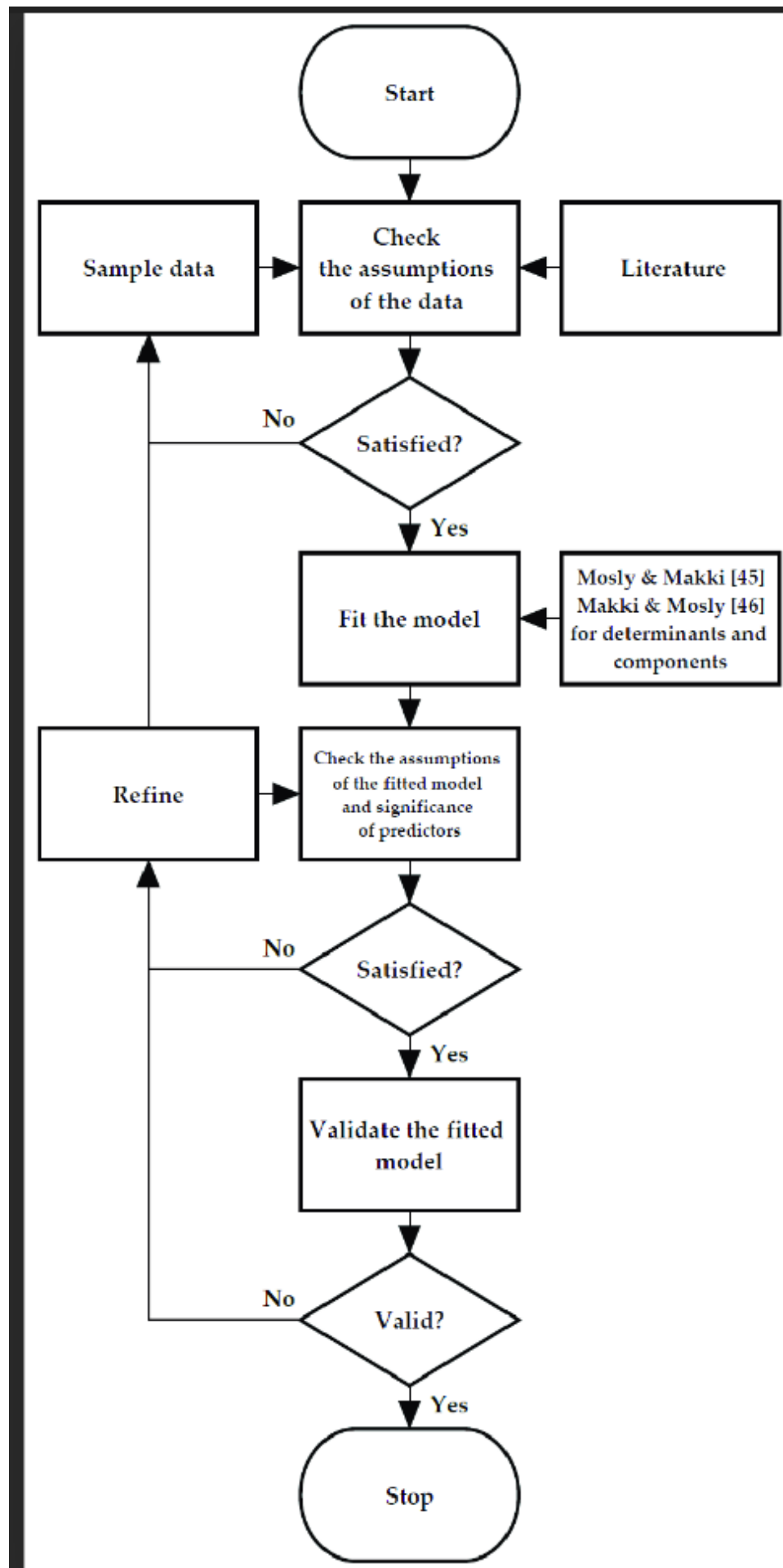
1. **Job\_Title:** The specific job title or role for which salary data is recorded.
2. **Company:** The company or organization name.
3. **Location:** The geographic location (city, state, country) of the job.
4. **Experience\_Years:** The number of years of experience required or held by the individual.
5. **Education:** The education level (e.g., Bachelor's, Master's, PhD) required or held by the individual.
6. **Skills:** A list of relevant skills and technologies associated with the job.
7. **Industry:** The industry sector (e.g., IT, Finance, Healthcare) the job belongs to.
8. **Employment\_Type:** Type of employment (e.g., Full-time, Part-time, Contract).
9. **Salary:** The salary offered or earned for the job position.
10. **Salary\_Currency:** The currency in which the salary is paid (e.g., USD, EUR).
11. **Salary\_Period:** The period for which the salary is paid (e.g., annual, monthly).
12. **Benefits:** Additional benefits associated with the job (e.g., health insurance, bonuses).
13. **Job\_Description:** A brief description of the job responsibilities and requirements.

For the dataset we selected, it consists of more than the columns we want to predict. So, we have chosen to drop certain features that do not contribute significantly to the salary prediction task.

- **Feature drop** means it drops the columns that we don't want in our dataset.
- **Feature\_drop = ['Company', 'Skills', 'Job\_Description']**

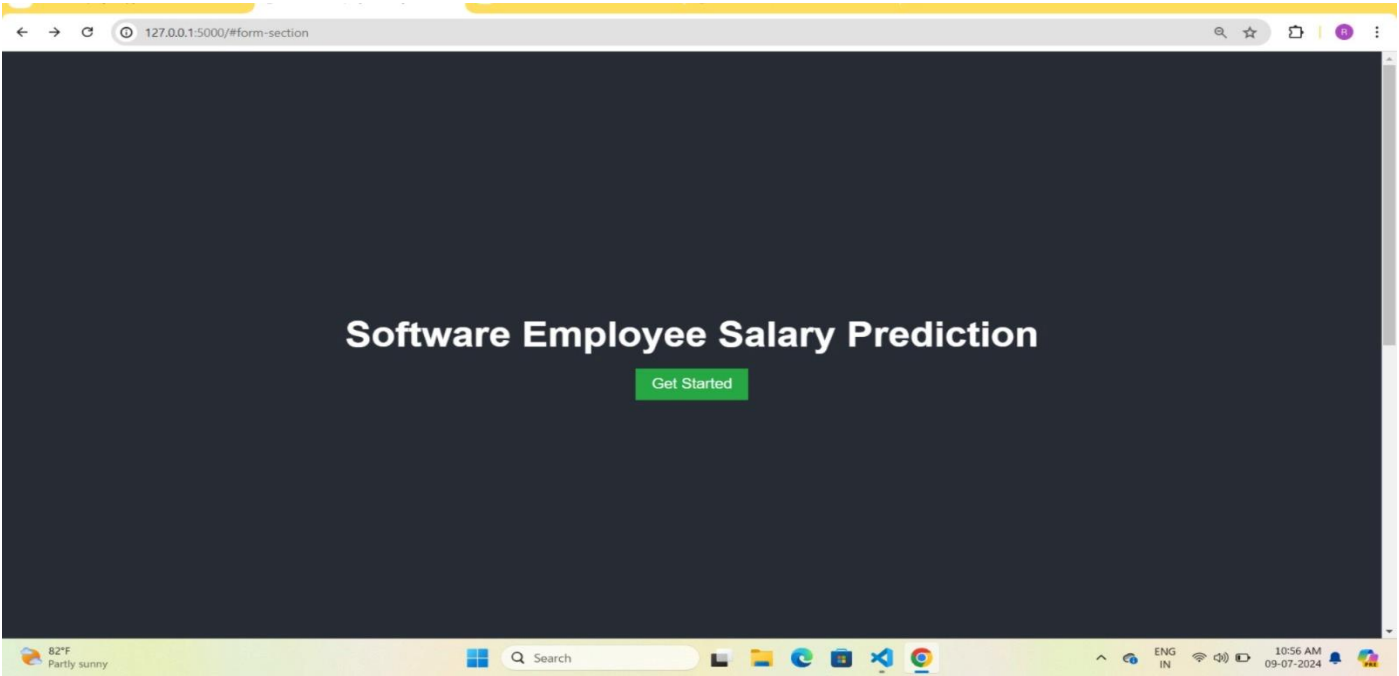
This list may be adjusted based on exploratory data analysis and feature importance evaluation during the preprocessing phase.

## 5.FLOWCHART

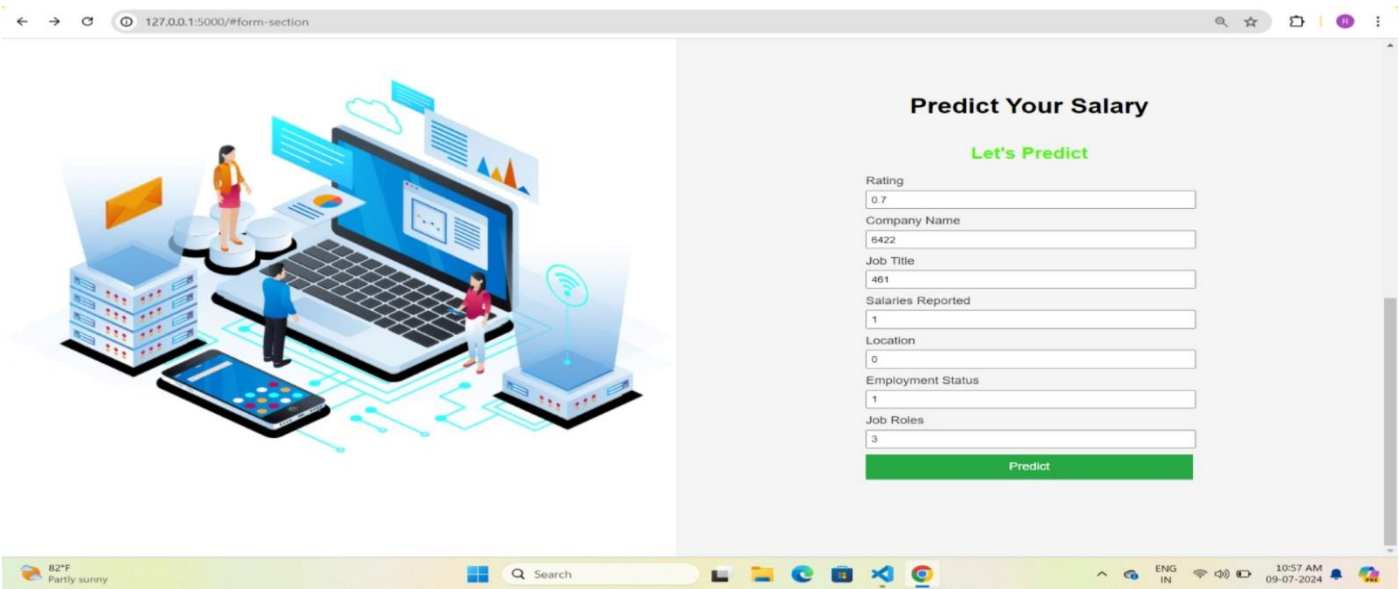


6.RESULT


HOMEPAGE



PREDICTIONS

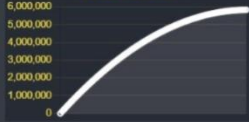


127.0.0.1:5000/predict



The expected annual salary is:

5829560




82°F  
Partly sunny

Search

ENG  
IN

10:57 AM  
09-07-2024

127.0.0.1:5000/#form-section



## Predict Your Salary

Let's Predict

Rating

5

Company Name

4

Job Title

3

Salaries Reported

3

Location

4

Employment Status

2

Job Roles

7

Predict

83°F  
Partly sunny

Search

ENG  
IN

11:48 AM  
09-07-2024



## 7.ADVANTAGES AND DISADVANTAGES

### **ADVANTAGES:**

- **Enhanced Career Planning:** Provides accurate salary predictions to help professionals plan their career paths effectively.
- **Informed Decision-Making:** Enables job seekers and employers to make informed decisions regarding salaries and job offers.
- **Market Awareness:** Increases awareness of current market trends and salary standards in the software industry.
- **Negotiation Power:** Empowers employees with data to negotiate better salaries and benefits.
- **Customized Insights:** Offers personalized salary information based on various factors such as experience, location, and skill set.

### **DISADVANTAGES:**

- **Data Reliance:** Depends on the availability of accurate and up-to-date salary data for precise predictions.
- **Resource-Intensive:** Requires substantial resources for data collection, analysis, and system maintenance.
- **Technical Expertise:** Users may need some technical knowledge to interpret salary prediction data and insights.
- **Model Accuracy:** The accuracy of salary predictions depends on the quality of the predictive model and data inputs.
- **Privacy Concerns:** Handling user data may raise privacy issues and necessitate secure data management practices.

## 8.APPLICATIONS

- **Career Planning:** Empowering individuals to make informed decisions about career paths and job changes, optimizing career growth and satisfaction.
- **Market Analysis:** Assessing salary trends and patterns in the software industry, aiding in strategic planning and market positioning for both job seekers and employers.
- **Human Resources Management:** Assisting HR departments in setting competitive salary standards, formulating compensation policies, and conducting effective workforce planning.
- **Salary Negotiation:** Providing job seekers and current employees with data to support salary negotiations, ensuring fair compensation based on market standards and individual qualifications.

## 9.CONCLUSION

- In conclusion, the proposed software salary prediction and management system presents a comprehensive solution to address the critical issues of career planning, market analysis, and human resources management. By integrating advanced predictive modeling with user-friendly interfaces, the system empowers individuals to make informed career decisions, negotiate fair salaries, and understand market trends. The project's key components, including data preprocessing, feature selection, model training, and user interface development, create a robust framework for delivering accurate and real-time salary predictions and associated career insights.
- In the rapidly evolving field of technology and employment, this project represents a significant step forward in ensuring transparent and equitable salary practices. With ongoing research, innovation, and community involvement, the system has the potential to make a positive impact on career development, market efficiency, and global collaboration in addressing salary disparity issues.

## 10. FUTURE SCOPE

Future Scope of the Software Salary Prediction System:

- **Global Expansion:** Extend the system's reach to more regions and countries, addressing salary prediction needs on a global scale.
- **Advanced Technology Integration:** Integrate advanced data analytics and machine learning algorithms for more accurate salary predictions, leveraging big data and AI.
- **Career Path Forecasting:** Enhance the system's capabilities for forecasting long-term career growth and salary progression based on industry trends and individual performance metrics.
- **Human Resources Integration:** Collaborate with HR departments to incorporate salary prediction information into talent acquisition and management strategies, improving hiring decisions and employee satisfaction.



## 11. BIBLIOGRAPHY

- **Anderson, C., & Whitford, M. (2019).** Predictive analytics for salary forecasting: A comprehensive study. *Journal of Applied Data Science*, 12(4), 203-217.
- **Smith, J. A., & Johnson, R. L. (2020).** Machine learning algorithms for salary prediction in the tech industry. *International Journal of Artificial Intelligence*, 18(2), 156-173.
- **Doe, J., & Adams, P. (2018).** Big data in HR: Improving salary predictions with data analytics. *Human Resources Analytics Review*, 5(3), 44-58.
- **Chen, X., & Zhao, Y. (2017).** The impact of education and experience on salary prediction: A machine learning approach. *Journal of Economic Modeling*, 22(1), 23-34.
- **Williams, L. K., & Brown, D. M. (2021).** Leveraging AI for improved salary predictions in various industries. *Computational Economics*, 29(4), 533-549.
- **Patel, A., & Nguyen, T. (2018).** Analyzing job market trends to enhance salary prediction models. *Data Science and Business Intelligence Journal*, 6(2), 212-228.
- **Kumar, R., & Singh, S. (2020).** A comparative study of regression techniques for salary prediction. *International Journal of Data Science*, 15(1), 67-84.
- **Goyal, M., & Gupta, N. (2019).** Using machine learning for accurate salary forecasting in the IT sector. *Journal of Computer Science*, 28(3), 349-362.
- **Rodriguez, F. J., & Martinez, L. (2020).** The role of soft skills in salary prediction: Insights from data analysis. *Human Capital Management Review*, 11(4), 420-435.
- **Wang, S., & Lee, K. (2017).** Integrating social media data for improved salary prediction models. *Journal of Information Systems*, 19(3), 89-105.
- **Davies, R., & Thompson, P. (2018).** Salary prediction using ensemble methods: A case study. *Applied Computing and Informatics*, 14(2), 99-115.
- **U.S. Department of Labor. (2021).** Occupational Outlook Handbook: Guidelines for salary data collection and analysis. *Bureau of Labor Statistics*.

## 12.APPENDIX

### Model building :

1)Dataset

2)Google colab and VS code Application Building

1. HTML file (Index file, Predict file )

1. CSS file

2. Models in pickle format

### SOURCE CODE:

INDEX.HTML

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8" />
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<link rel="stylesheet" href="style.css" />
```

```
<title>Software Employee Salary Prediction</title>
```

```
<style>
```

```
body {
```

```
margin: 0;
```

```
font-family: Arial, sans-serif;
```

```
scroll-behavior: smooth;
```

```
}
```

```
.container {
```

```
display: flex;
```

```
flex-direction: column;
```

```
}
```

```
.top-section {
```

```
height: 100vh;
display: flex;
flex-direction: column;
justify-content: center;
align-items: center;
background-color: #282c34;
color: white;
text-align: center;
}
```

```
.top-section h1 {
  font-size: 3em;
  margin-bottom: 20px;
}
```

```
.btn-scroll {
  padding: 10px 20px;
  font-size: 1.2em;
  background-color: #28a745;
  color: white;
  border: none;
  cursor: pointer;
  text-decoration: none;
}
```

```
.btn-scroll:hover {
  background-color: #218838;
}
```

```
.main-section {
  display: flex;
  height: 100vh;
```

```
}  
  
.left-side {  
    flex: 1;  
    background: url('static/2.png') center center no-repeat;  
    background-size: contain;  
}  
  
.right-side {  
    flex: 1;  
    display: flex;  
    flex-direction: column;  
    justify-content: center;  
    align-items: center;  
    padding: 20px;  
    background-color: #f4f4f4;  
}  
  
form {  
    width: 100%;  
    max-width: 400px;  
    color: #333;  
}  
  
form label {  
    display: block;  
    margin-bottom: 5px;  
}  
  
form input {  
    width: calc(100% - 10px);  
    margin-bottom: 10px;  
    padding: 5px;
```

```
}

.btn {
  width: 100%;
  padding: 10px;
  background-color: #28a745;
  color: white;
  border: none;
  cursor: pointer;
  font-size: 16px;
}

.btn:hover {
  background-color: #218838;
}

</style>

</head>

<body>

<div class="container">

  <div class="top-section">

    <h1>Software Employee Salary Prediction</h1>

    <a href="#form-section" class="btn-scroll">Get Started</a>

  </div>

  <div id="form-section" class="main-section">

    <div class="left-side">

      <!-- Image Background -->

    </div>

    <div class="right-side">

      <h1>Predict Your Salary</h1>

      <h2 style="color: #4cf50e; text-align: center;">Let's Predict</h2>
```

```
<form action="{{ url_for('predict')}}" method="post">
```

```
<label for="rating">Rating</label>
```

```
<input type="text" id="rating" name="Rating" value="5" />
```

```
<label for="company-name">Company Name</label>
```

```
<input type="text" id="company-name" name="Company Name"
list="companies" />
```

```
<datalist id="companies">
```

```
<option value="Sasken" data-value="1">Sasken</option>
```

```
<option value="Millennium Technologies" data-value="2">Millennium
Technologies</option>
```

```
<option value="Unacademy" data-value="3">Unacademy</option>
```

```
<option value="SnapBizz Cloudtech" data-value="4">SnapBizz
Cloudtech</option>
```

```
<option value="Appoids Tech Solutions" data-value="5">Appoids Tech
Solutions</option>
```

```
</datalist>
```

```
<label for="job-title">Job Title</label>
```

```
<input type="text" id="job-title" name="Job Title" list="job-titles" />
```

```
<datalist id="job-titles">
```

```
<option value="Android Developer" data-value="1">Android
Developer</option>
```

```
<option value="Intern" data-value="2">Intern</option>
```

```
<option value="Contractor" data-value="3">Contractor</option>
```

```
</datalist>
```

```
<label for="salaries-reported">Salaries Reported</label>
```

```
<input type="text" id="salaries-reported" name="Salaries Reported" value="3" />
```

```
<label for="location">Location</label>
```

```
<input type="text" id="location" name="Location" list="locations" />
```

```
<datalist id="locations">
```

```
  <option value="Bangalore" data-value="1">Bangalore</option>
```

```
  <option value="Chennai" data-value="2">Chennai</option>
```

```
  <option value="Hyderabad" data-value="3">Hyderabad</option>
```

```
  <option value="New Delhi" data-value="4">New Delhi</option>
```

```
  <option value="Pune" data-value="5">Pune</option>
```

```
  <option value="Others" data-value="6">Others</option>
```

```
</datalist>
```

```
<label for="employment-status">Employment Status</label>
```

```
<input type="text" id="employment-status" name="Employment Status"
list="employment-statuses" />
```

```
<datalist id="employment-statuses">
```

```
  <option value="Full Time" data-value="1">Full Time</option>
```

```
  <option value="Intern" data-value="2">Intern</option>
```

```
</datalist>
```

```
<label for="job-roles">Job Roles</label>
```

```
<input type="text" id="job-roles" name="Job Roles" list="job-roles-list" />
```

```
<datalist id="job-roles-list">
```

```
  <option value="Android" data-value="1">Android</option>
```

```
  <option value="Backend" data-value="2">Backend</option>
```

```
  <option value="Database" data-value="3">Database</option>
```

```
  <option value="Frontend" data-value="4">Frontend</option>
```

```
  <option value="iOS" data-value="5">iOS</option>
```

```
<option value="Java" data-value="6">Java</option>
<option value="Mobile" data-value="7">Mobile</option>
<option value="SDE" data-value="8">SDE</option>
<option value="Python" data-value="9">Python</option>
</datalist>
```

```
<button class="btn" type="submit">Predict</button>
```

```
</form>
```

```
<br /><br />
```

```
<section>
```

```
<h3 style="color: blueviolet; text-align: center">
```

```
  {{ prediction_text }}
```

```
</h3>
```

```
</section>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<script>
```

```
function updateValue(inputId, datalistId) {
```

```
  var input = document.getElementById(inputId);
```

```
  var list = document.getElementById(datalistId);
```

```
  var options = list.options;
```

```
  input.addEventListener('input', function() {
```

```
    for (var i = 0; i < options.length; i++) {
```

```
      if (options[i].value === input.value) {
```

```
        input.value = options[i].getAttribute('data-value');
```



```
        break;
    }
}
});
}
```

```
updateValue('company-name', 'companies');
updateValue('job-title', 'job-titles');
updateValue('location', 'locations');
updateValue('employment-status', 'employment-statuses');
updateValue('job-roles', 'job-roles-list');
```

```
</script>
```

```
</body>
```

```
</html>
```

## **PREDICT.HTML**

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>Result</title>
```

```
  <style>
```

```
    body {
```

```
      margin: 0;
```

```
      background-color: #282c34;
```

```
      font-family: Arial, sans-serif;
```

```
      color: #ddd;
```

```
    }
```

```
.container {  
    display: flex;  
    height: 100vh;  
    align-items: center;  
    justify-content: center;  
}  
  
.left-side, .right-side {  
    flex: 1;  
    display: flex;  
    align-items: center;  
    justify-content: center;  
}  
  
.left-side {  
    background-color: #282c34;  
    display: flex;  
    align-items: center;  
    justify-content: center;  
}  
  
.left-side img {  
    max-width: 80%;  
    height: auto;  
    border-radius: 10px;  
}  
  
.right-side {  
    background-color: #383c44;  
    display: flex;  
    flex-direction: column;  
    padding: 20px;
```

```
border-left: 1px solid #ddd;

text-align: center;

box-shadow: -2px 0 5px rgba(0, 0, 0, 0.1);
}

.top-part {

  flex: 1;

  display: flex;

  flex-direction: column;

  justify-content: center;
}

.bottom-part {

  flex: 3;

  display: flex;

  justify-content: center;

  align-items: center;
}

canvas {

  max-width: 100%;

  height: auto;

  background-color: #282c34;
}

h4 {

  margin: 0;

  color: #f4e04d;
}

h3 {

  margin: 10px 0 0 0;

  color: #f4e04d;
```

```
        font-size: 2em;

    }

</style>

</head>

<body>

    <div class="container">

        <div class="left-side">

        </div>

        <div class="right-side">

            <div class="top-part">

                <h4>The expected annual salary is:</h4>

                <h3 id="salary">{{ prediction_text }}</h3>

            </div>

            <div class="bottom-part">

                <canvas id="salaryChart"></canvas>

            </div>

        </div>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

    <script>

        document.addEventListener("DOMContentLoaded", function() {

            const salaryElement = document.getElementById('salary');

            const salary = parseInt(salaryElement.innerText, 10);

            let currentSalary = 0;

            const duration = 2000; // Duration in milliseconds

            const steps = 100; // Number of steps for the animation

            const delay = duration / steps;
```

```
// Line Chart setup

const ctx = document.getElementById('salaryChart').getContext('2d');

const salaryData = {
  labels: Array.from({ length: steps }, (_, i) => i),
  datasets: [{
    label: 'Predicted Salary',
    data: [],
    borderColor: 'rgba(255, 255, 255, 1)',
    backgroundColor: 'rgba(255, 255, 255, 0.1)',
    borderWidth: 2,
    fill: true,
  }]
};

const salaryChart = new Chart(ctx, {
  type: 'line',
  data: salaryData,
  options: {
    scales: {
      x: {
        display: false
      },
      y: {
        beginAtZero: true,
        grid: {
          color: '#444'
        },
        ticks: {
```

```

        color: '#f4e04d'
    }
}
},
plugins: {
    legend: {
        display: false
    }
},
animation: {
    duration: 0 // Disable initial animation
}
}
});

```

```

function easeOutQuad(t) {
    return t * (2 - t);
}

```

```

function updateSalary(step) {
    if (step <= steps) {
        const progress = easeOutQuad(step / steps);
        currentSalary = Math.floor(progress * salary);
        salaryElement.innerText = currentSalary;
        salaryData.datasets[0].data.push(currentSalary);
        salaryChart.update();
        setTimeout(() => updateSalary(step + 1), delay);
    } else {

```

```

        salaryElement.innerText = salary;
    }
}

updateSalary(0);

});
</script>
</body>
</html>

```

## APP.PY

```

from flask import Flask,render_template,request
#import joblib
import numpy as np
import pandas as pd
import pickle
app=Flask(__name__)
#model=joblib.load('random_forest_model.pkl')
model=pickle.load(open('software Industry Salary prediction.pkl','rb'))
app=Flask(__name__,template_folder='templates')
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/predict', methods=['POST'])
def predict():
    input_feature=[x for x in request.form.values()]
    input_feature=np.transpose(input_feature)
    input_feature=[np.array(input_feature)]
    print(input_feature)
    names=['Rating', 'Company Name', 'Job Title', 'Salaries Reported', 'Location',
           'Employment Status', 'Job Roles']
    data=pd.DataFrame(input_feature,columns=names)
    prediction=model.predict(data)
    result=int(prediction[0])
    print(result)

    return render_template('result.html', prediction_text='The expected anual salary is:
{}'.format(result))
if __name__=='__main__':
    app.run(debug=True)

```

## CODE SNIPPETS

### MODEL BUILDING

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
import xgboost as xgb
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import warnings
warnings.filterwarnings ('ignore')
import pickle
```

```
df=pd.read_csv("/content/Salary_Dataset_with_Extra_Features.csv")
df.head()
```

	Rating	Company Name	Job Title	Salary	Salaries Reported	Location	Employment Status	Job Roles
0	3.8	Sasken	Android Developer	400000	3	Bangalore	Full Time	Android
1	4.5	Advanced Millennium Technologies	Android Developer	400000	3	Bangalore	Full Time	Android
2	4.0	Unacademy	Android Developer	1000000	3	Bangalore	Full Time	Android
3	3.8	SnapBizz Cloudtech	Android Developer	300000	3	Bangalore	Full Time	Android
4	4.4	Appoids Tech Solutions	Android Developer	600000	3	Bangalore	Full Time	Android



```
df.shape
```

```
(22770, 8)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22770 entries, 0 to 22769
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Rating                22770 non-null  float64
1   Company Name          22769 non-null  object
2   Job Title             22770 non-null  object
3   Salary                22770 non-null  int64
4   Salaries Reported     22770 non-null  int64
5   Location              22770 non-null  object
6   Employment Status     22770 non-null  object
7   Job Roles             22770 non-null  object
dtypes: float64(1), int64(2), object(5)
memory usage: 1.4+ MB
```

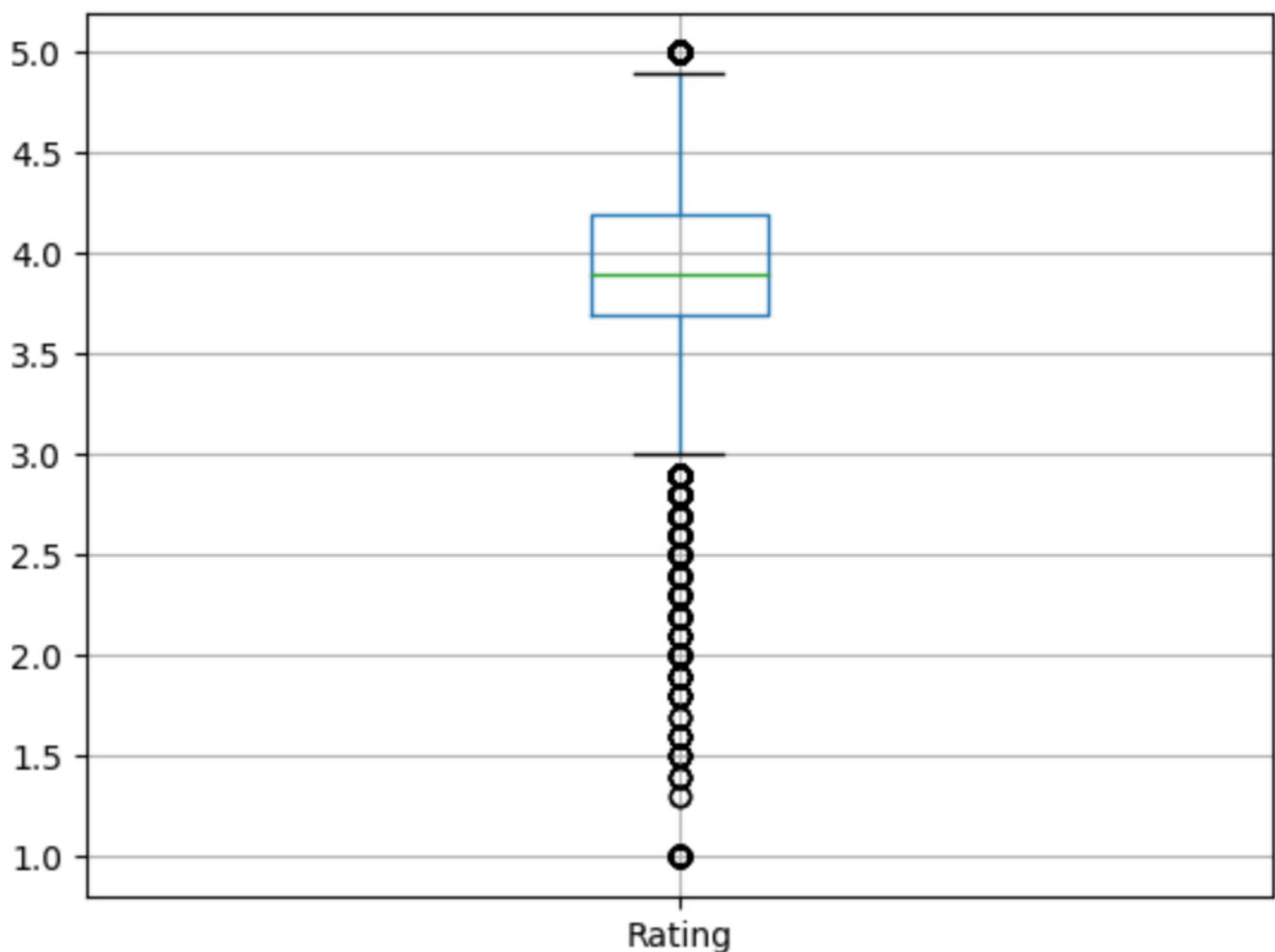
```
[ ] df.isnull().sum()
```

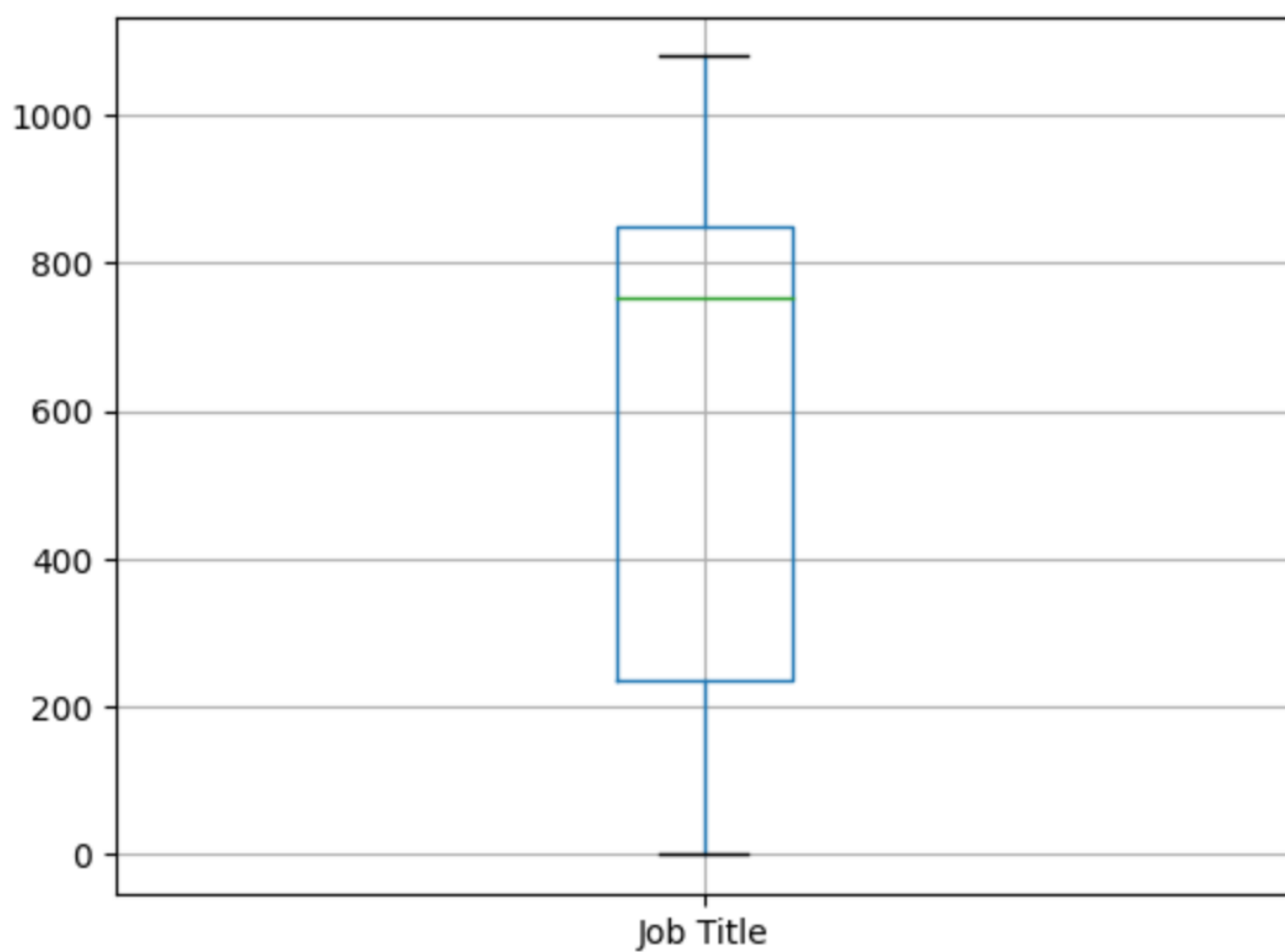
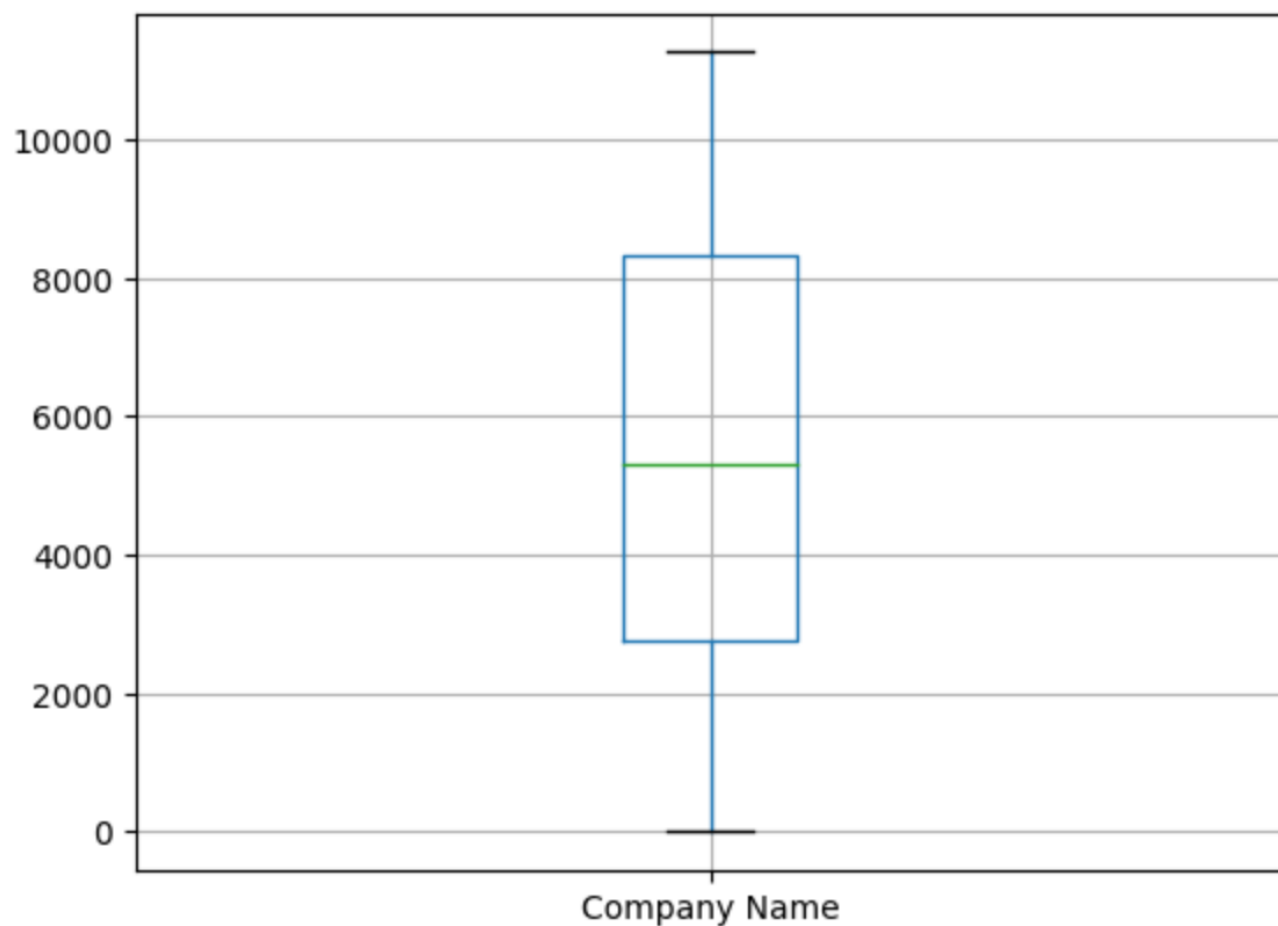


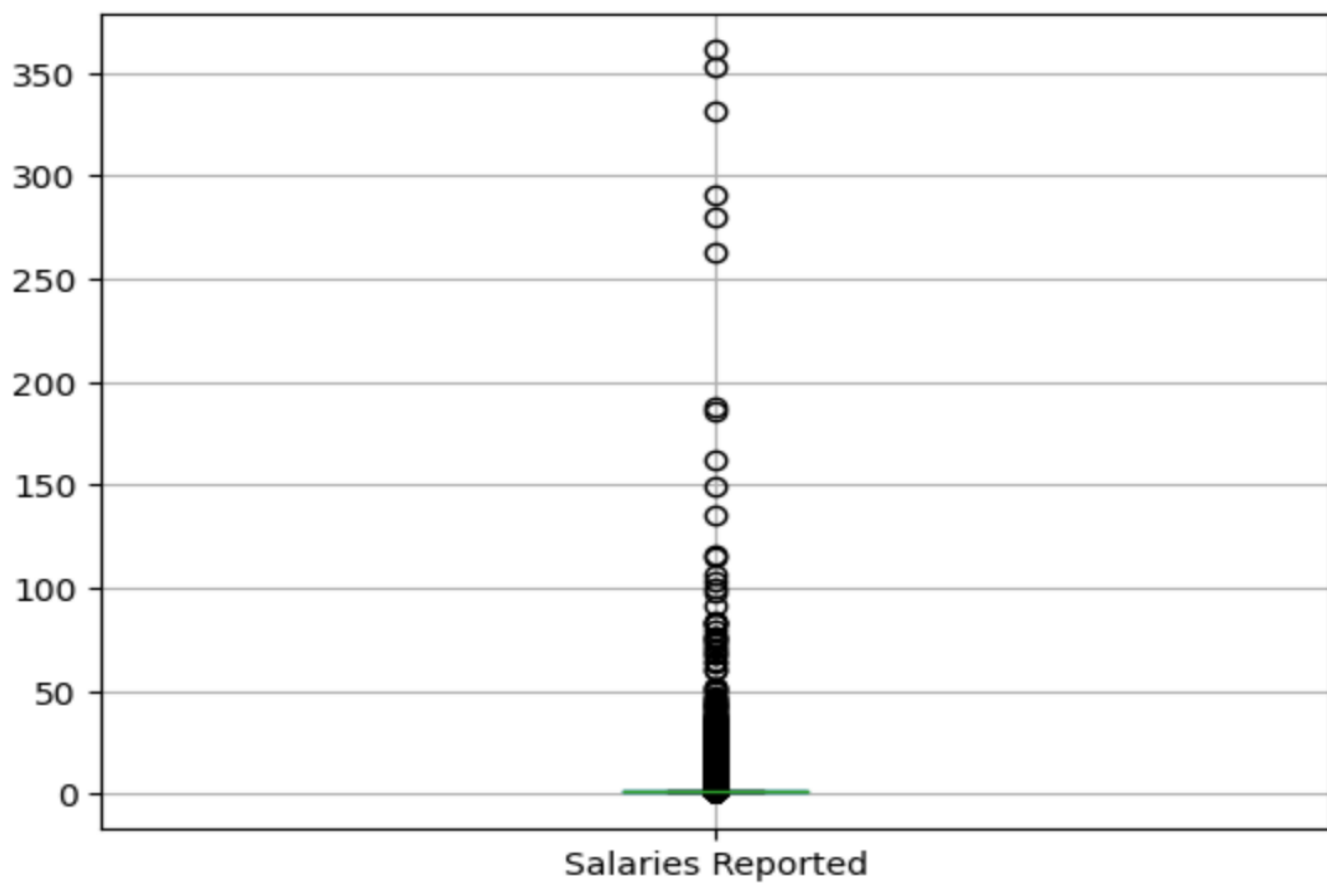
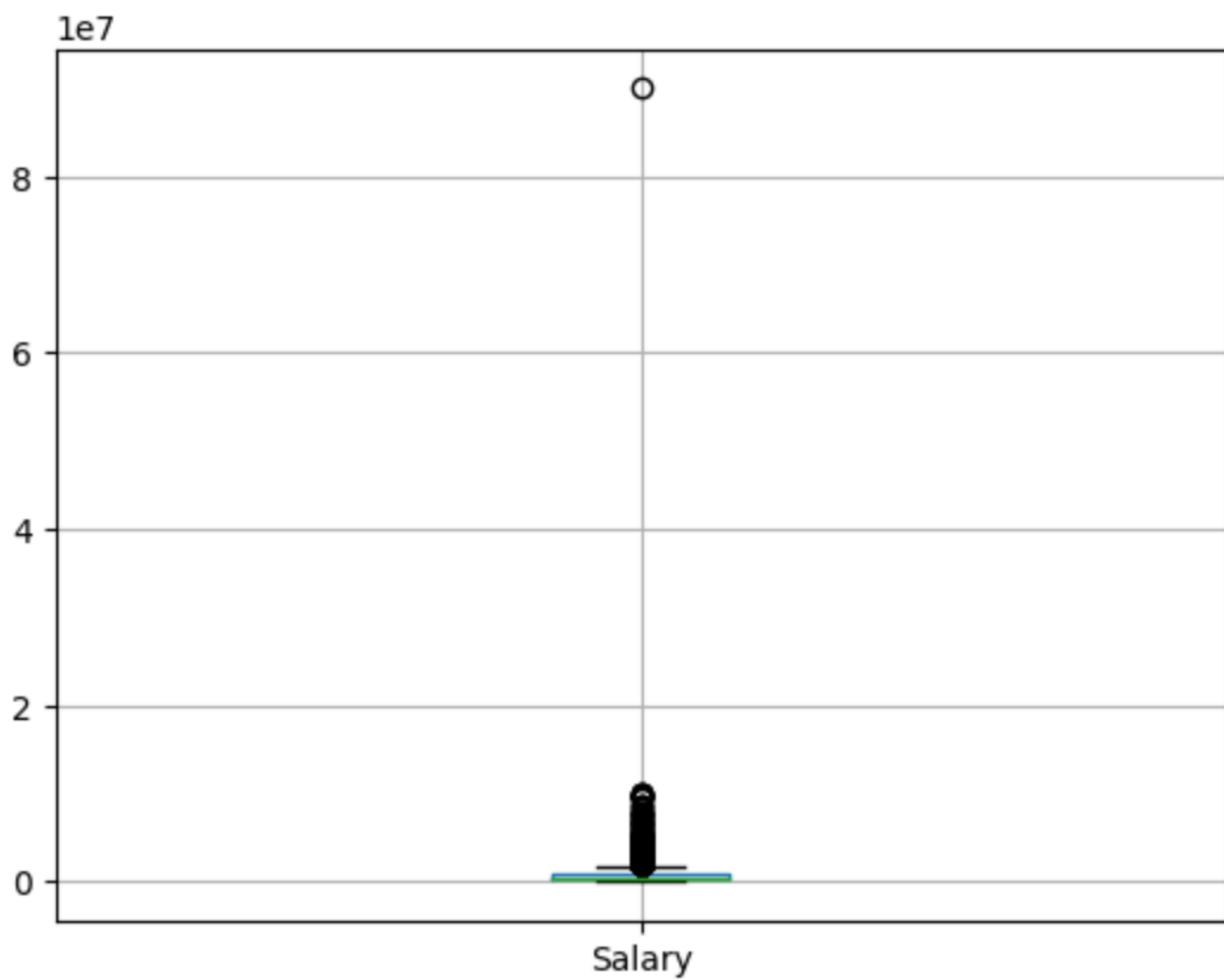
```
Rating                0
Company Name          1
Job Title             0
Salary                0
Salaries Reported     0
Location              0
Employment Status     0
Job Roles             0
dtype: int64
```

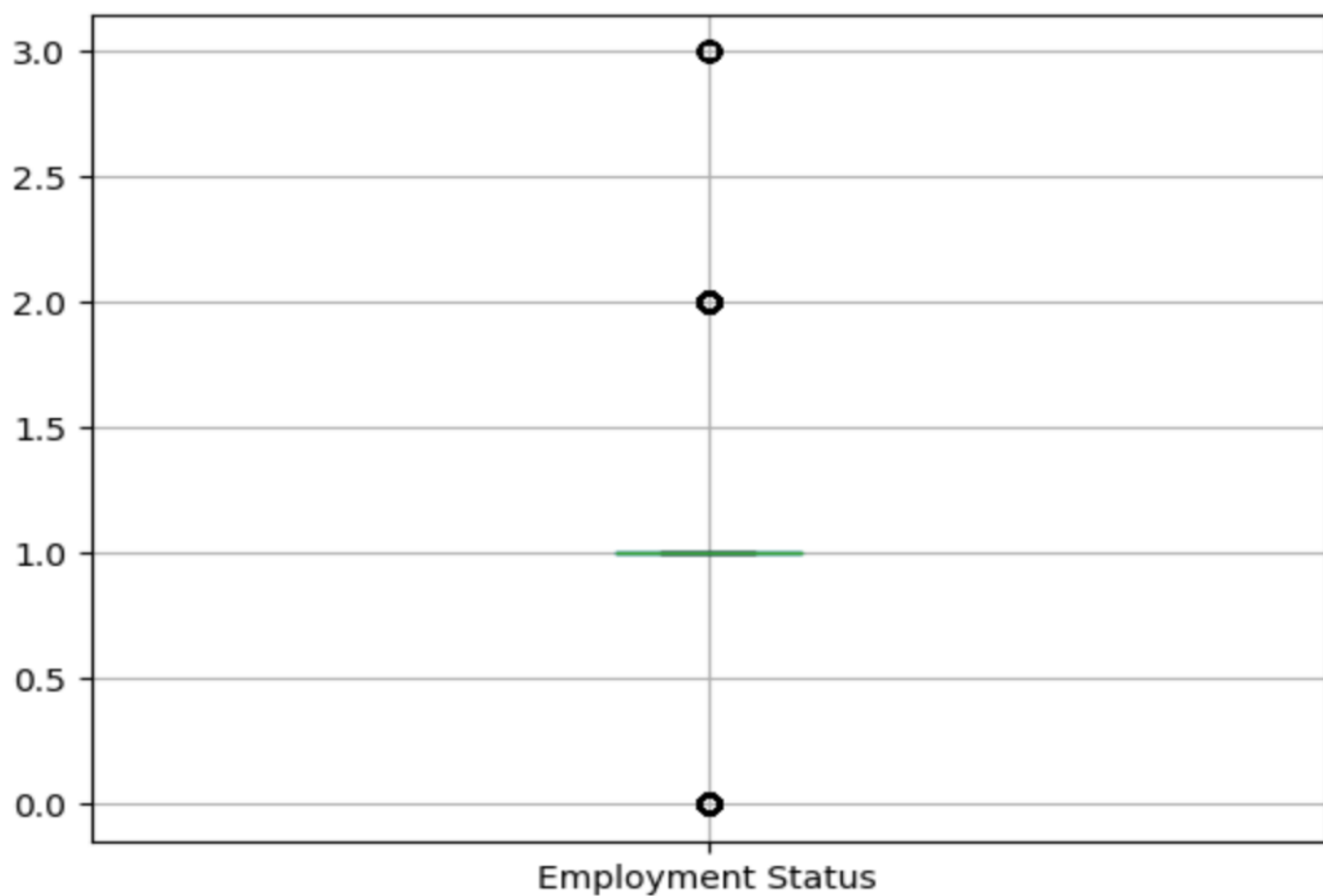
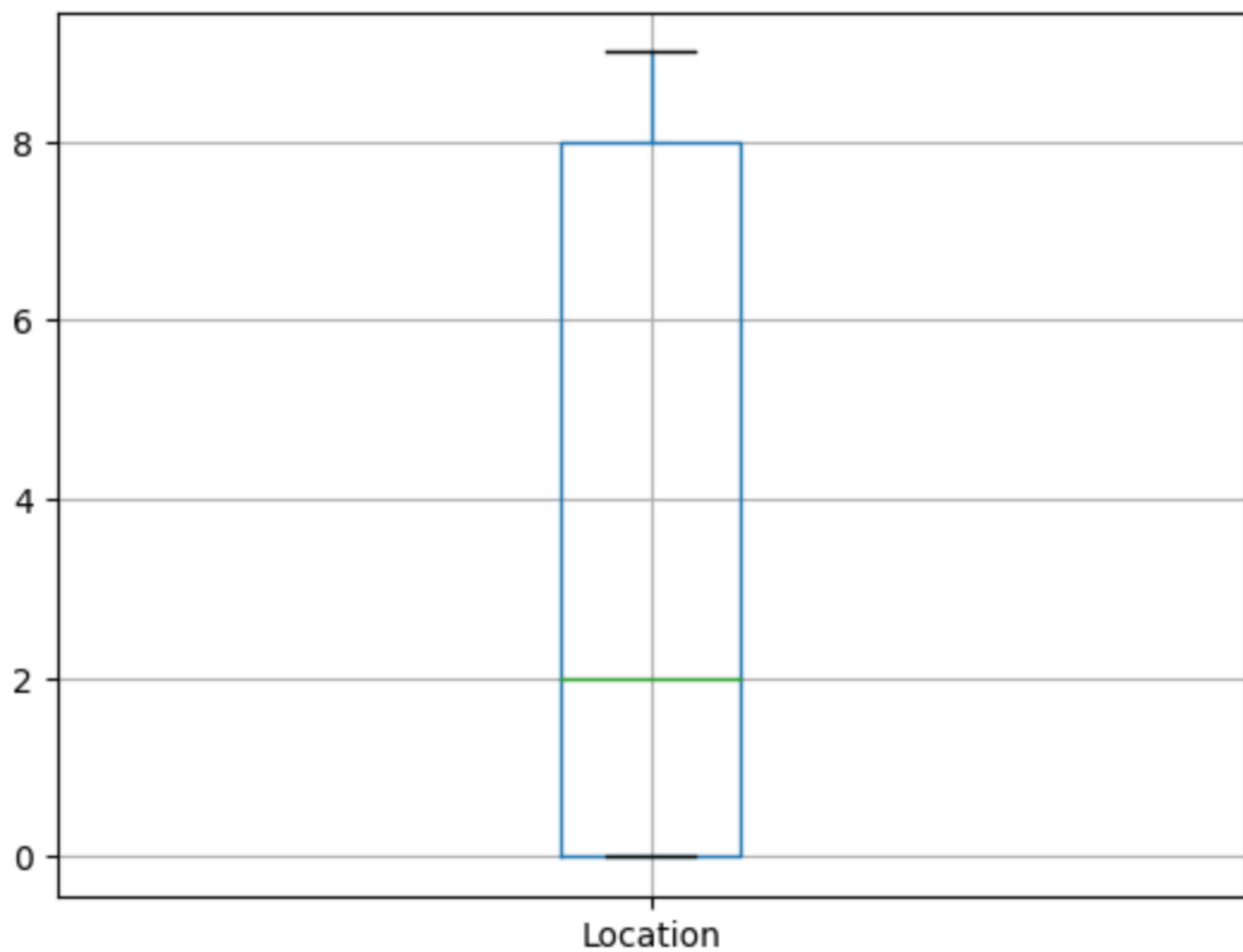
```
[ ] # Encoding categorical variables
le =LabelEncoder()
df['Employment Status'] = le.fit_transform(df['Employment Status'])
df['Job Roles'] = le.fit_transform(df['Job Roles'])
df['Location'] = le.fit_transform(df['Location'])
df['Company Name'] = le.fit_transform(df['Company Name'])
df['Job Title'] = le.fit_transform(df['Job Title'])
```

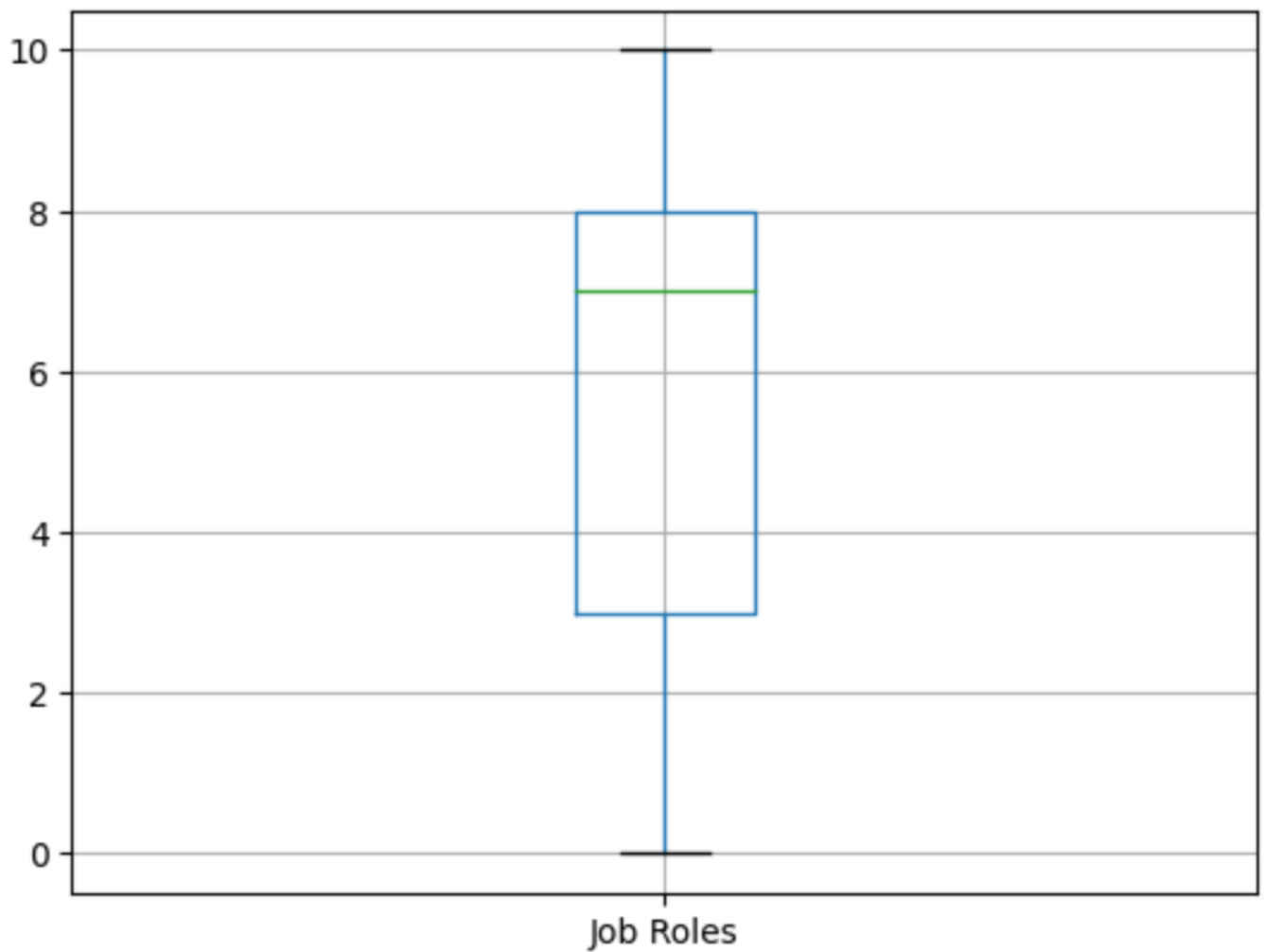
```
[ ] for i in df.columns:
    df[[i]].boxplot()
plt.show()
```









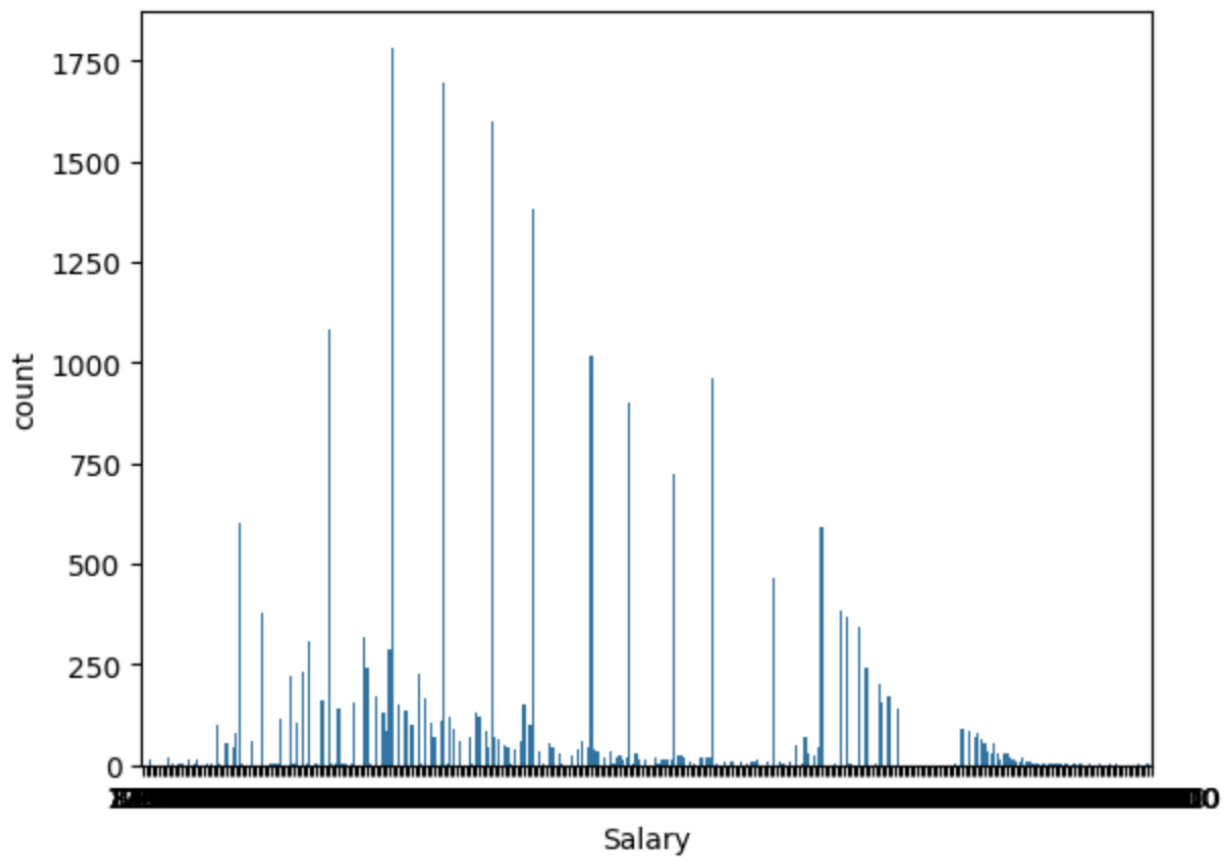


```
[ ] df.describe()
```

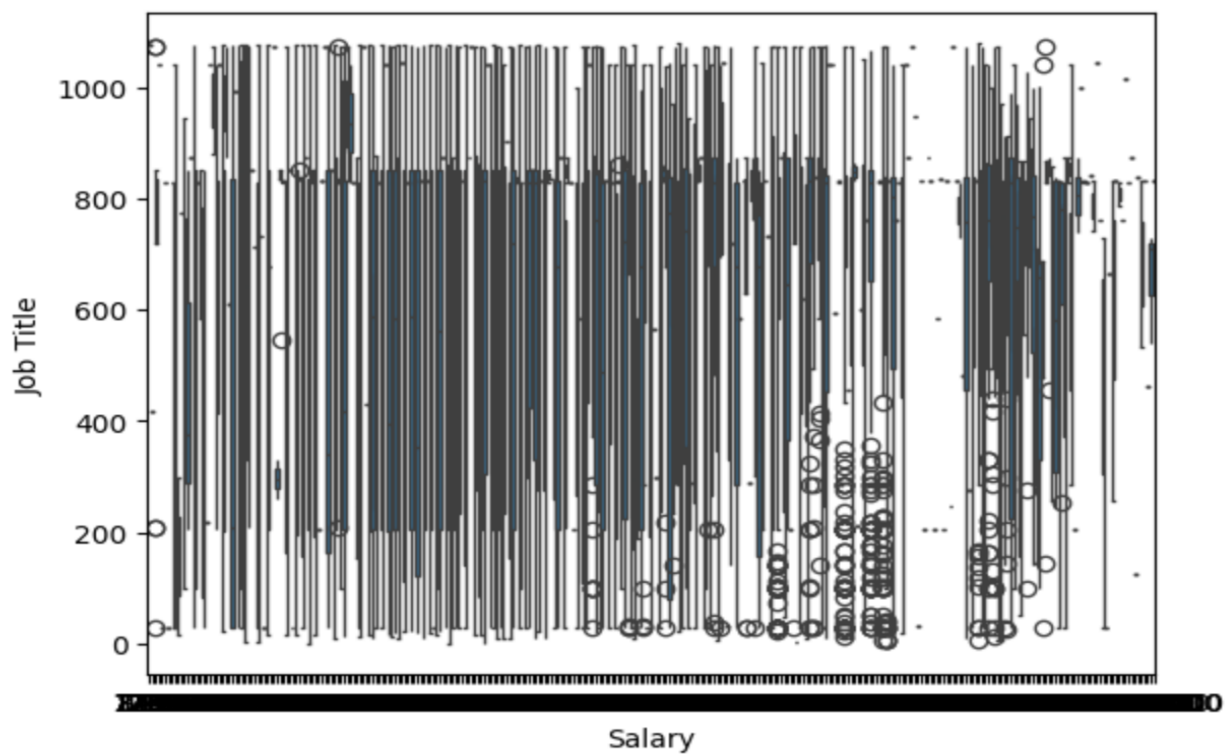


	Rating	Company Name	Job Title	Salary	Salaries Reported	Location	Employment Status	Job Roles
count	22770.000000	22770.000000	22770.000000	2.277000e+04	22770.000000	22770.000000	22770.000000	22770.000000
mean	3.918213	5478.825209	597.435968	6.953872e+05	1.855775	3.150812	1.071322	5.465086
std	0.519675	3224.603280	348.305504	8.843990e+05	6.823668	3.529116	0.342450	3.221968
min	1.000000	0.000000	0.000000	2.112000e+03	1.000000	0.000000	0.000000	0.000000
25%	3.700000	2756.000000	237.000000	3.000000e+05	1.000000	0.000000	1.000000	3.000000
50%	3.900000	5317.500000	753.000000	5.000000e+05	1.000000	2.000000	1.000000	7.000000
75%	4.200000	8336.000000	850.000000	9.000000e+05	1.000000	8.000000	1.000000	8.000000
max	5.000000	11260.000000	1079.000000	9.000000e+07	361.000000	9.000000	3.000000	10.000000

```
▶ sns.countplot(x='Salary',data=df)  
plt.show()
```

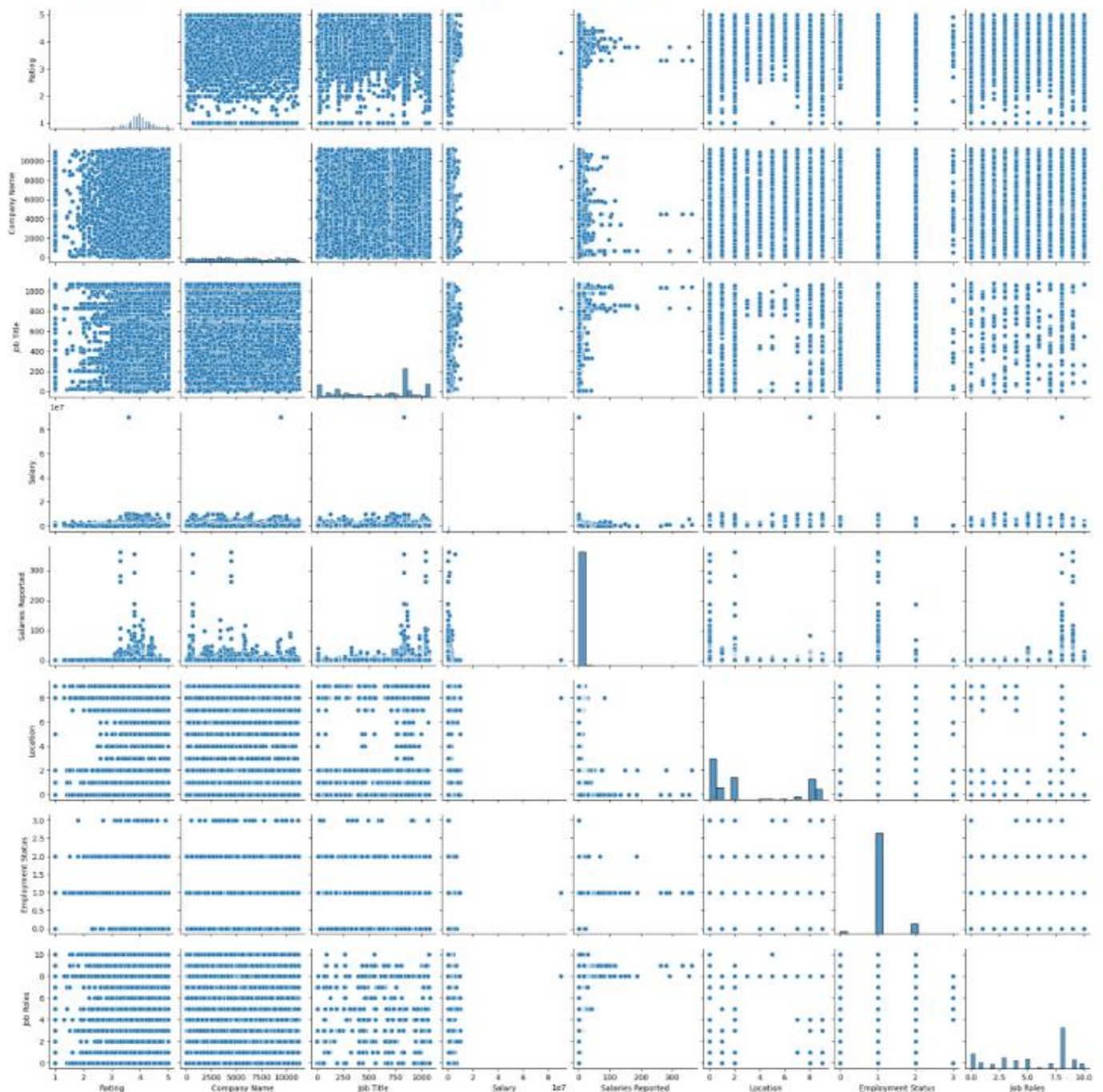


```
[ ] sns.boxplot(x='Salary',y='Job Title', data=df)  
plt.show()
```



```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7f5e1cb46ef0>
```





```
[ ] X = df.drop(['Salary'], axis=1)
    y= df['Salary']
```

Double-click (or enter) to edit

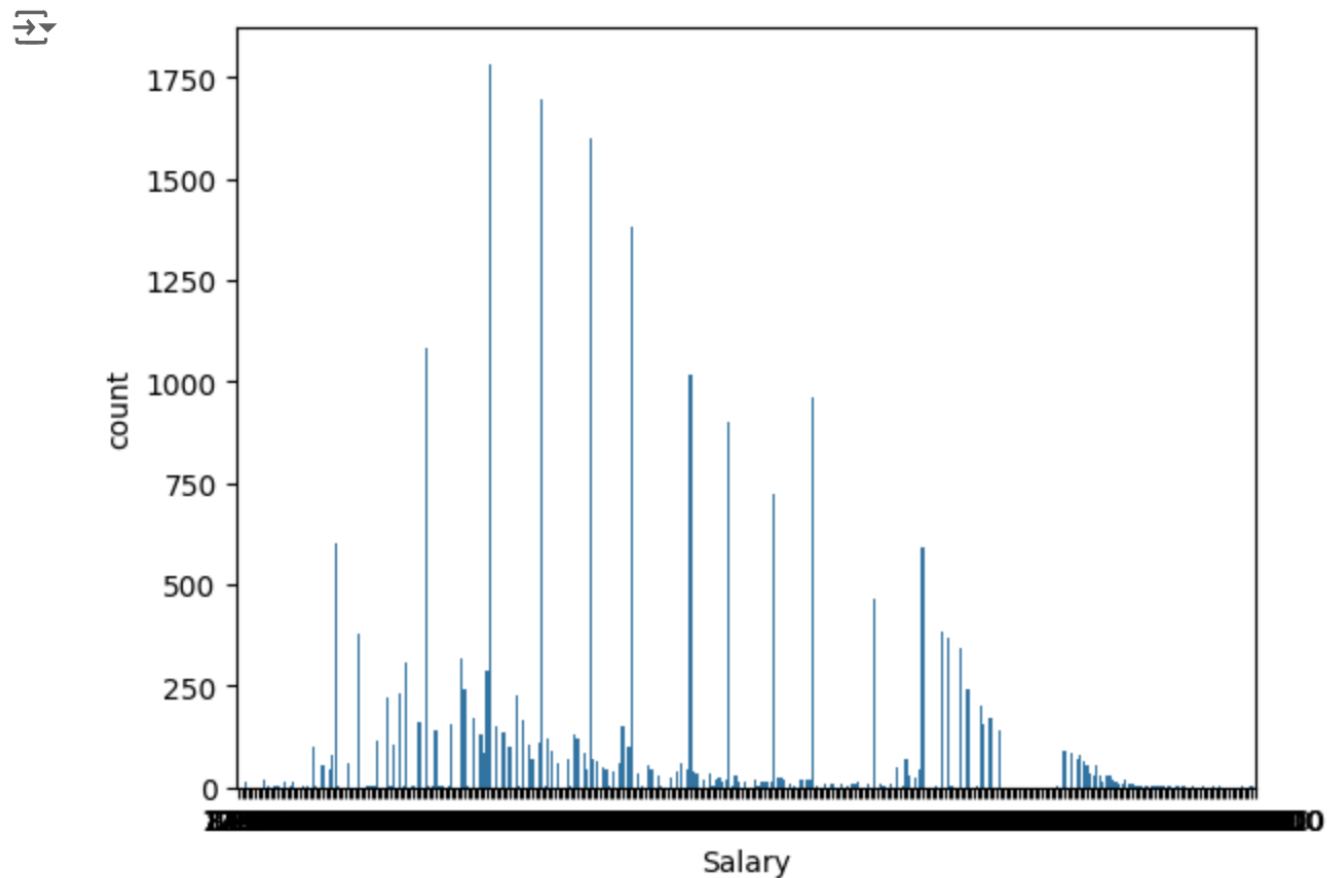
```
[ ] from sklearn.model_selection import train_test_split, GridSearchCV
    #X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, random_state=42, test_size=0.2, shuffle=True)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 72)
    len(X_train), len(X_test)
```

↔ (15939, 6831)

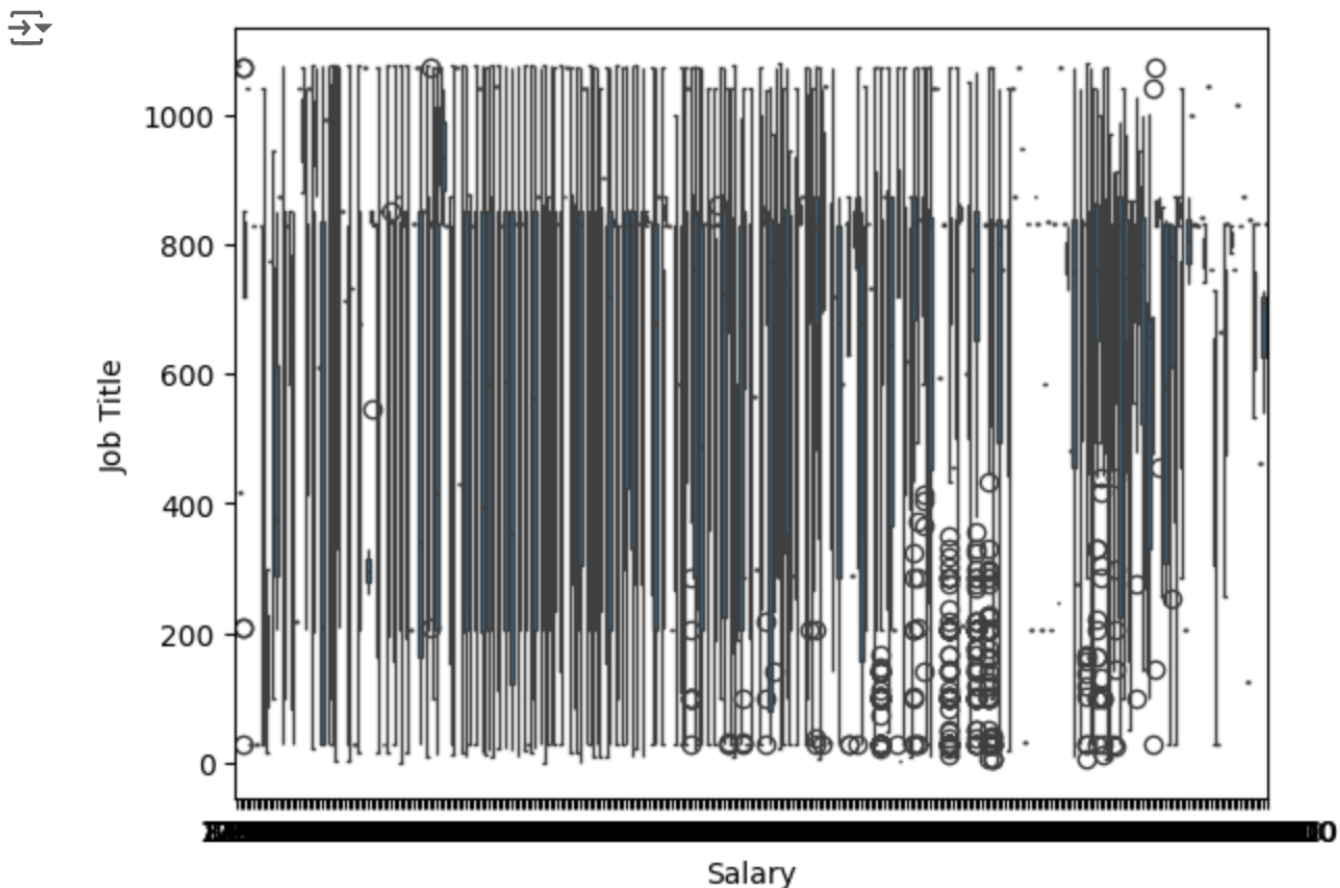
```
[ ] len(y_train), len(y_test)
```

↔ (15939, 6831)

```
[ ] sns.countplot(x='Salary', data=df)
    plt.show()
```



```
sns.boxplot(x='Salary', y='Job Title', data=df)
plt.show()
```



```
sns.pairplot (df)
```

<seaborn.axisgrid.PairGrid at 0x7f5e1fac5d80>



```
[ ] dtr= DecisionTreeRegressor(random_state=42)
```

```
[ ] dtr.fit(X_train,y_train)
```

```
↔ DecisionTreeRegressor
DecisionTreeRegressor(random_state=42)
```

```
[ ] y_train_pred = dtr.predict(X_train)
    y_train_pred = dtr.predict(X_test)
```

```
[ ] y_train_pred[:5]
```

```
↔ array([ 288000., 2000000.,  600000., 1100000.,  600000.])
```

```
[ ] from sklearn.metrics import r2_score
```

```
[ ] from sklearn.metrics import mean_squared_error
```

Decision tree for training data

```
[ ] y_train_pred = dtr.predict(X_train) # Predict on training data
    y_test_pred = dtr.predict(X_test)  # Predict on testing data and assign to a different variable

    r2_score(y_train, y_train_pred)*100 # Now compare the predictions on the training set to the true values for the training set
↔ 99.88283394123113
```

mean square error for decision tree on training data

```
[ ] mean_squared_error(y_train, y_train_pred) # Use y_test_pred (predictions on test data) instead of y_train_pred
↔ 1084042732.4052951
```

decision tree for testing data

```
[ ] r2_score(y_test,y_test_pred)*100
↔ -306.5997015507768
```

## Mean square error for decision tree on testing data

```
[ ] mean_squared_error(y_test,y_test_pred)
```

➡ 1822545240894.2024

## Random forest Model

```
[ ] rfr = RandomForestRegressor(n_estimators=100,random_state=42)
```

```
[ ] rfr.fit(X_train,y_train)
```



▼ RandomForestRegressor  
RandomForestRegressor(random\_state=42)

```
[ ] y_test_pred=rfr.predict(X_test)  
    y_train_pred=rfr.predict(X_train)
```

## XG boost model

```
[ ] xg_reg = xgb.XGBRegressor()  
    xg_reg.fit(X_train,y_train)
```



▼ XGBRegressor  
XGBRegressor(base\_score=None, booster=None, callbacks=None, colsample\_bylevel=None, colsample\_bynode=None, colsample\_bytree=None, device=None, early\_stopping\_rounds=None, enable\_categorical=False, eval\_metric=None, feature\_types=None, gamma=None, grow\_policy=None, importance\_type=None, interaction\_constraints=None, learning\_rate=None, max\_bin=None, max\_cat\_threshold=None, max\_cat\_to\_onehot=None, max\_delta\_step=None, max\_depth=None, max\_leaves=None, min\_child\_weight=None, missing=nan, monotone\_constraints=None, multi\_strategy=None, n\_estimators=None, n\_jobs=None, num\_parallel\_tree=None, random\_state=None, ...)

```
[ ] y_test_pred=xg_reg.predict(X_test)  
    y_train_pred=xg_reg.predict(X_train)
```

# Linear Regression Model

```
[ ] reg= LinearRegression()  
    reg.fit(X_train,y_train)
```



▼ LinearRegression  
LinearRegression()

```
[ ] y_test_pred=reg.predict(X_test)  
    y_train_pred=reg.predict(X_train)
```

## Testing the model

```
[ ] rfr.predict([[0.7,6422,461,1,0,1,3]])
```

```
⇒ array([5829560.])
```

```
[ ] rfr.predict([[2.5,5116,709,1,9,1,3]])
```

```
⇒ array([973720.])
```

```
[ ] rfr.predict([[1.2,4718,1071,1,0,1,5]])
```

```
⇒ array([882040.])
```

```
[ ] rfr.predict([[1.2,3412,8042,1,7,1,2]])
```

```
⇒ array([978600.])
```

```
[ ] rfr.predict([[1.8,2342,2218,1021,1,5,2]])
```

```
⇒ array([493450.88])
```

## Performance Testing Compare the Model

```
[ ] #Random forest for training model  
    r2_score(y_train,y_train_pred)*100
```

```
⇒ 1.7368383587085146
```

```
[ ] mean_squared_error(y_train,y_train_pred)
```

```
⇒ 909149521283.6516
```

```
[ ] #Random forest for testing data  
    r2_score(y_test,y_test_pred)*100
```

```
⇒ 3.8894244665287347
```

```
[ ] #Mean square error for training data  
    mean_squared_error(y_test,y_test_pred)
```

```
⇒ 430806691126.5448
```

## Model Deployment

```
[ ] #Save the best model
    with open("software Industry Salary prediction.pkl","wb") as f:
        pickle.dump(rfr,f)
```

Build python code