Software Salary Prediction Using Machine Learning

Team Id: LTVIP2025TMID41628

By:

Team Leader: Shaik Manisha

Team member: Goli Varun Kumar

Team member: Mandala Pujitha

Team member: Bathula Kalyan Babu

Abstract

This project focuses on predicting salaries based on various features such as company name, company rating, job title, salaries reported ,employment status and location using machine learning algorithms. The model is trained on a dataset of salary records and is integrated with a web-based interface developed using Flask. The aim is to provide an intelligent system that can estimate salaries for different roles in the industry, assisting both recruiters and job seekers.

Table of Contents

1. Introduction

In the modern job market, salary estimation plays an essential role for employees and employers alike. With the advancement of Artificial Intelligence (AI) and Machine Learning (ML), data-driven predictions can now be made with high accuracy. This project implements a salary prediction system that utilizes ML algorithms trained on real-world data. The web interface allows users to input their details and receive an estimated salary instantly.

2. Problem Statement

Determining fair and accurate salaries is often challenging due to the diversity of job roles, industries, and experience levels. Manual methods or generic online tools may not consider all influencing factors. This project addresses this issue by creating a machine learning-based model that predicts salaries using multiple relevant features, providing more personalized and data-driven results.

3. Objectives

- To collect and preprocess a dataset containing various job-related features.

- To train and evaluate ML models for accurate salary prediction.

- To develop a Flask web application for user interaction.

- To visualize the dataset and model performance for better understanding.

4. Existing System vs Proposed System

Existing System:

Existing salary estimation methods rely mainly on static data or manual entry systems, which are often inaccurate. These systems do not utilize machine learning and thus lack predictive capabilities.

Proposed System:

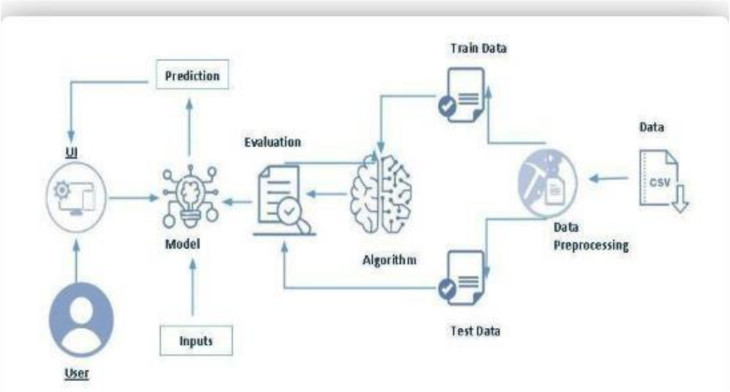
The proposed system leverages machine learning techniques to predict salaries dynamically. It considers multiple features simultaneously and improves accuracy through training and evaluation.

5. System Architecture

The system consists of three primary components:

1. Data Collection and Preprocessing
2. Model Training and Evaluation
3. Flask Web Interface for Deployment

Technical Architecture:



6. Modules Description

The project is divided into the following modules:

1. Data Preprocessing Module

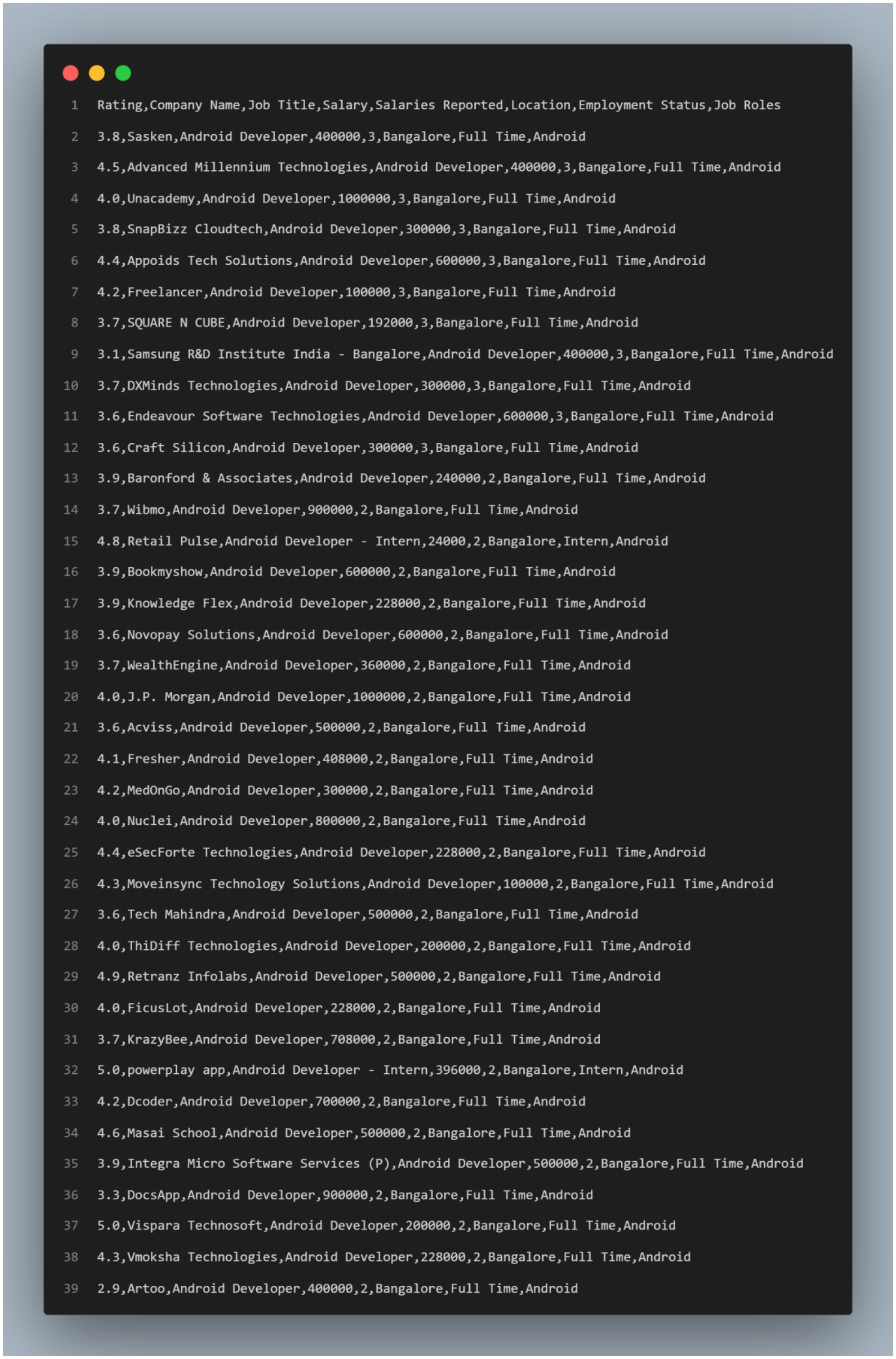
2. Model Training Module

3. Web Application Module

4. Result and Visualization Module

7. Dataset Description

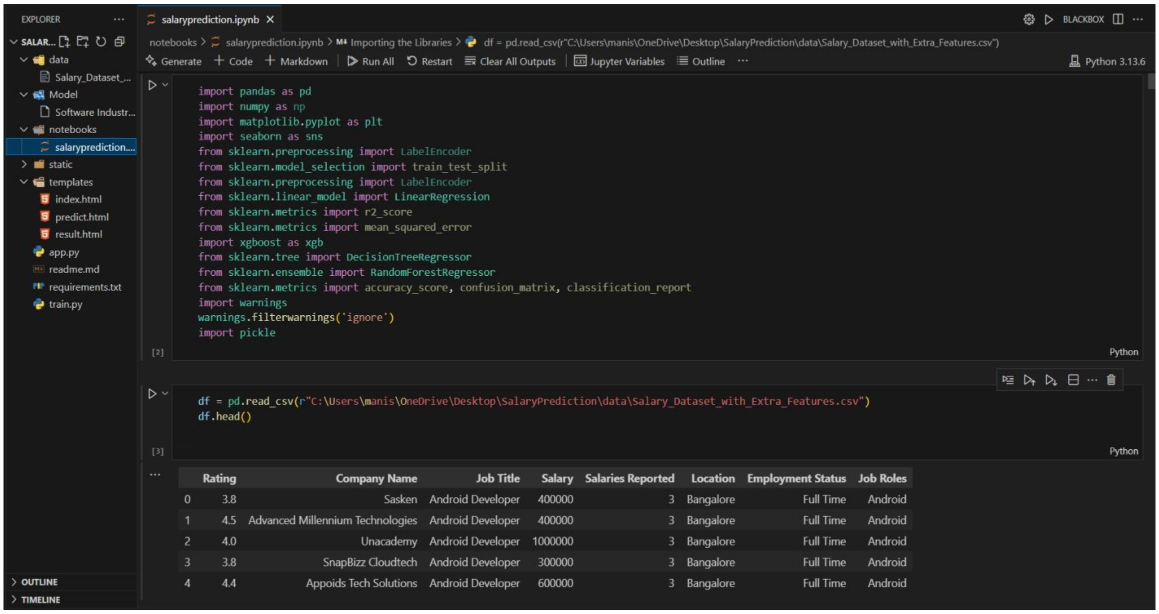
The dataset used in this project 'Salary_Dataset_with_Extra_Features.csv' contains features like company name, rating, job title, salaries reported, location, employment status and job roles These attributes significantly influence the salary outcome. The data was cleaned, encoded, and standardized before model training.



```
Rating,Company Name,Job Title,Salary,Salaries Reported,Location,Employment Status,Job Roles
3.8,Sasken,Android Developer,400000,3,Bangalore,Full Time,Android
4.5,Advanced Millennium Technologies,Android Developer,400000,3,Bangalore,Full Time,Android
4.0,Unacademy,Android Developer,1000000,3,Bangalore,Full Time,Android
3.8,SnapBizz Cloudtech,Android Developer,300000,3,Bangalore,Full Time,Android
4.4,Appoids Tech Solutions,Android Developer,600000,3,Bangalore,Full Time,Android
4.2,Freelancer,Android Developer,100000,3,Bangalore,Full Time,Android
3.7,SQUARE N CUBE,Android Developer,192000,3,Bangalore,Full Time,Android
3.1,Samsung R&D Institute India - Bangalore,Android Developer,400000,3,Bangalore,Full Time,Android
3.7,DXMinds Technologies,Android Developer,300000,3,Bangalore,Full Time,Android
3.6,Endeavour Software Technologies,Android Developer,600000,3,Bangalore,Full Time,Android
3.6,Craft Silicon,Android Developer,300000,3,Bangalore,Full Time,Android
3.9,Baronford & Associates,Android Developer,240000,2,Bangalore,Full Time,Android
3.7,Wibmo,Android Developer,900000,2,Bangalore,Full Time,Android
4.8,Retail Pulse,Android Developer - Intern,24000,2,Bangalore,Intern,Android
3.9,Bookmyshow,Android Developer,600000,2,Bangalore,Full Time,Android
3.9,Knowledge Flex,Android Developer,228000,2,Bangalore,Full Time,Android
3.6,Novopay Solutions,Android Developer,600000,2,Bangalore,Full Time,Android
3.7,WealthEngine,Android Developer,360000,2,Bangalore,Full Time,Android
4.0,J.P. Morgan,Android Developer,1000000,2,Bangalore,Full Time,Android
3.6,Acviss,Android Developer,500000,2,Bangalore,Full Time,Android
4.1,Fresher,Android Developer,408000,2,Bangalore,Full Time,Android
4.2,MedOnGo,Android Developer,300000,2,Bangalore,Full Time,Android
4.0,Nuclei,Android Developer,800000,2,Bangalore,Full Time,Android
4.4,eSecForte Technologies,Android Developer,228000,2,Bangalore,Full Time,Android
4.3,Moveinsync Technology Solutions,Android Developer,100000,2,Bangalore,Full Time,Android
3.6,Tech Mahindra,Android Developer,500000,2,Bangalore,Full Time,Android
4.0,ThiDiff Technologies,Android Developer,200000,2,Bangalore,Full Time,Android
4.9,Retranz Infolabs,Android Developer,500000,2,Bangalore,Full Time,Android
4.0,FicusLot,Android Developer,228000,2,Bangalore,Full Time,Android
3.7,KrazyBee,Android Developer,708000,2,Bangalore,Full Time,Android
5.0,powerplay app,Android Developer - Intern,396000,2,Bangalore,Intern,Android
4.2,Dcoder,Android Developer,700000,2,Bangalore,Full Time,Android
4.6,Masai School,Android Developer,500000,2,Bangalore,Full Time,Android
3.9,Integra Micro Software Services (P),Android Developer,500000,2,Bangalore,Full Time,Android
3.3,DocsApp,Android Developer,900000,2,Bangalore,Full Time,Android
5.0,Vispara Technosoft,Android Developer,200000,2,Bangalore,Full Time,Android
4.3,Vmoksha Technologies,Android Developer,228000,2,Bangalore,Full Time,Android
2.9,Artoo,Android Developer,400000,2,Bangalore,Full Time,Android
```

8. Model Training and Evaluation

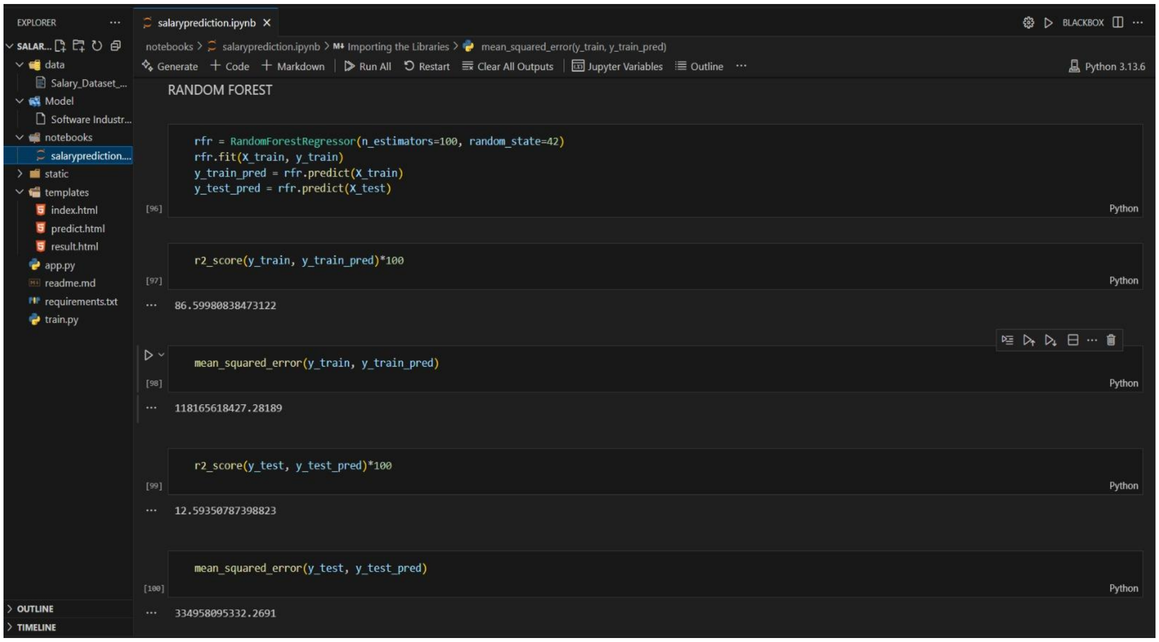
The training process involves loading the dataset, preprocessing, splitting into training and testing sets, and applying algorithms such as Random Forest Regressor for prediction. The model's performance was evaluated based on metrics like Mean Squared Error (MSE) and R^2 Score.



```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
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.metrics import accuracy_score, confusion_matrix, classification_report
import warnings
warnings.filterwarnings('ignore')
import pickle
```

```python
df = pd.read_csv(r"C:\Users\manis\OneDrive\Desktop\SalaryPrediction\data\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



RANDOM FOREST

```python
rfr = RandomForestRegressor(n_estimators=100, random_state=42)
rfr.fit(X_train, y_train)
y_train_pred = rfr.predict(X_train)
y_test_pred = rfr.predict(X_test)
```

```python
r2_score(y_train, y_train_pred)*100
```

```
86.59980838473122
```

```python
mean_squared_error(y_train, y_train_pred)
```

```
118165618427.28189
```

```python
r2_score(y_test, y_test_pred)*100
```

```
12.59350787398823
```

```python
mean_squared_error(y_test, y_test_pred)
```

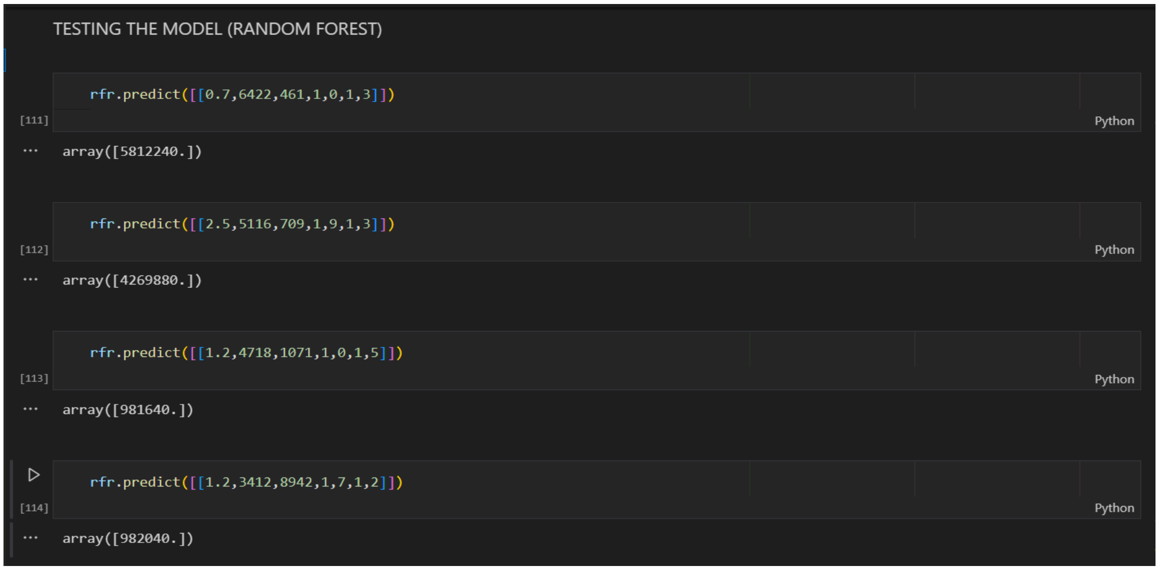
```
334958095332.2691
```

9. Implementation

The implementation was carried out using Python. Flask framework was used for backend web development, and HTML/CSS for frontend design. The trained ML model was integrated into Flask using Pickle serialization. Users can enter their job-related details on the web page to get predicted salary values.

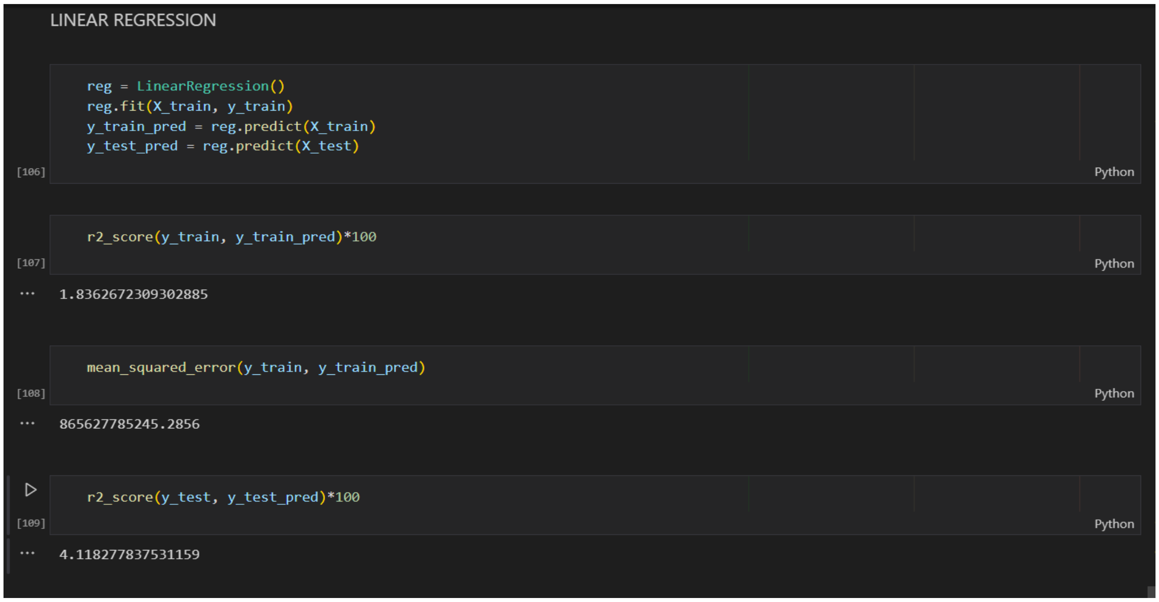
10. Results

The system successfully predicts salary values based on user input. The results show that the Random Forest model provides accurate predictions compared to other algorithms tested. The web interface allows seamless interaction with the model.



TESTING THE MODEL (RANDOM FOREST)

```
[111] rfr.predict([[0.7,6422,461,1,0,1,3]])
array([5812240.])

[112] rfr.predict([[2.5,5116,709,1,9,1,3]])
array([4269880.])

[113] rfr.predict([[1.2,4718,1071,1,0,1,5]])
array([981640.])

[114] rfr.predict([[1.2,3412,8942,1,7,1,2]])
array([982040.])
```



LINEAR REGRESSION

```
[106]
reg = LinearRegression()
reg.fit(X_train, y_train)
y_train_pred = reg.predict(X_train)
y_test_pred = reg.predict(X_test)

[107] r2_score(y_train, y_train_pred)*100
1.8362672309302885

[108] mean_squared_error(y_train, y_train_pred)
865627785245.2856

[109] r2_score(y_test, y_test_pred)*100
4.118277837531159
```

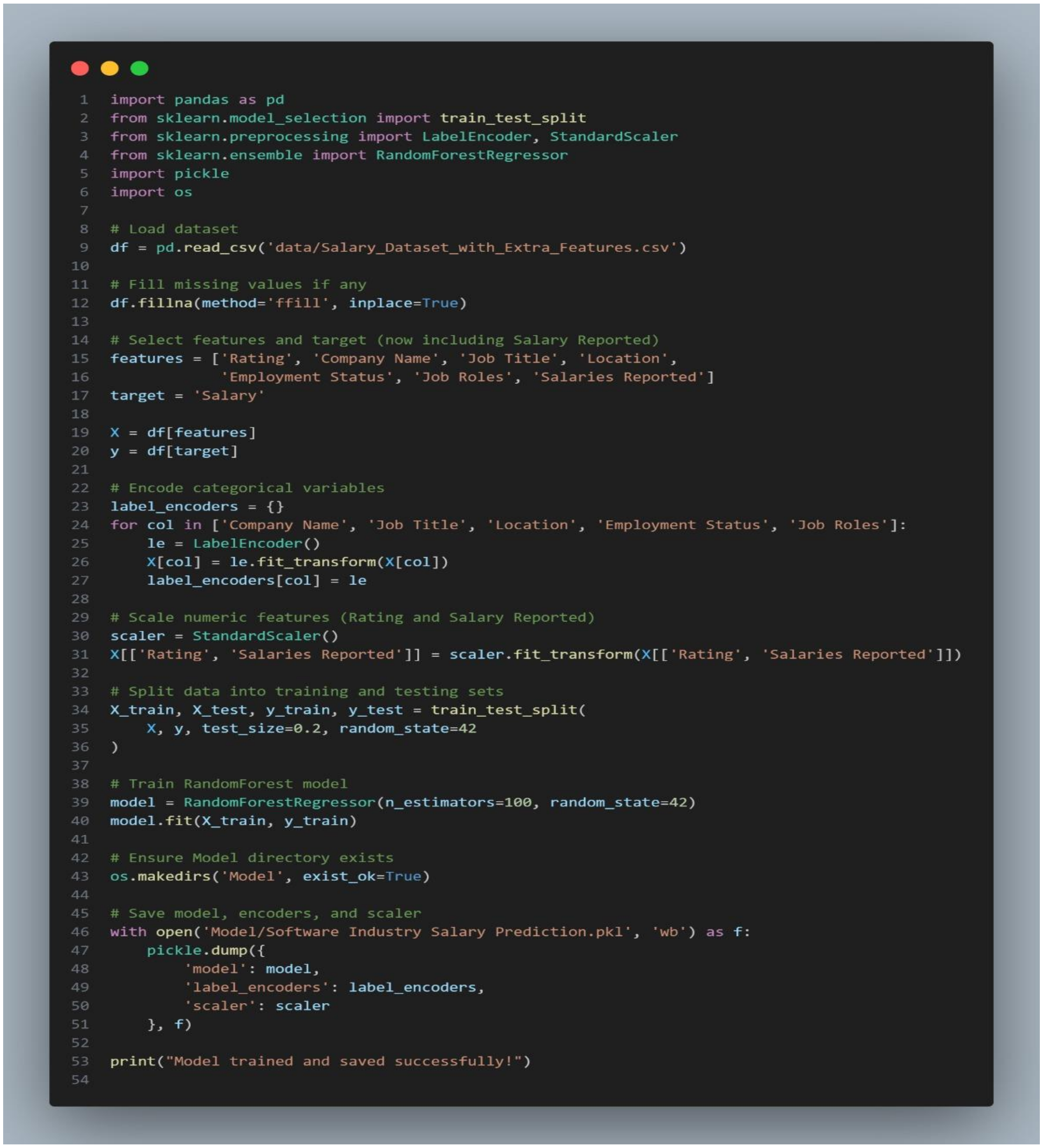
11. Conclusion and Future Scope

This project demonstrates how machine learning can be applied to real-world problems like salary prediction. The model achieves good accuracy and helps users estimate expected salaries effectively. Future enhancements may include expanding the dataset, adding more features and integrating deep learning methods for improved predictions.

9.1 Demonstration

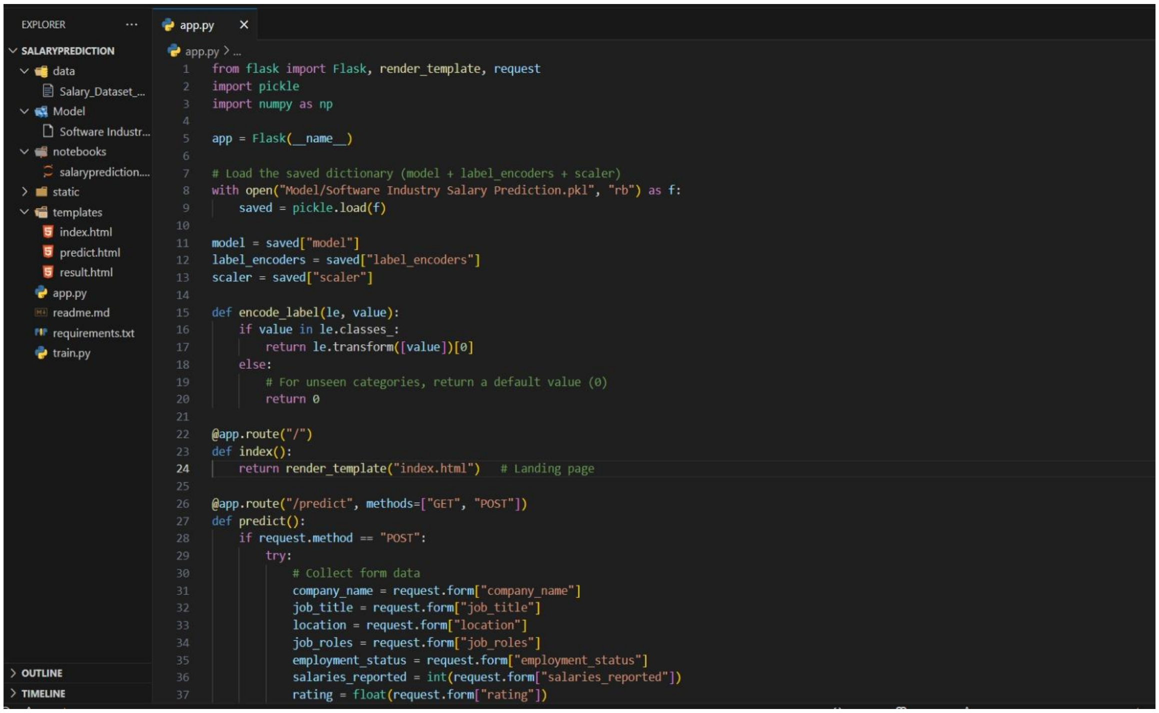
This section provides visual evidence of the developed project, including source code snippets and the web application interface.

Figure 1: Screenshot of train.py (Model Training Code)

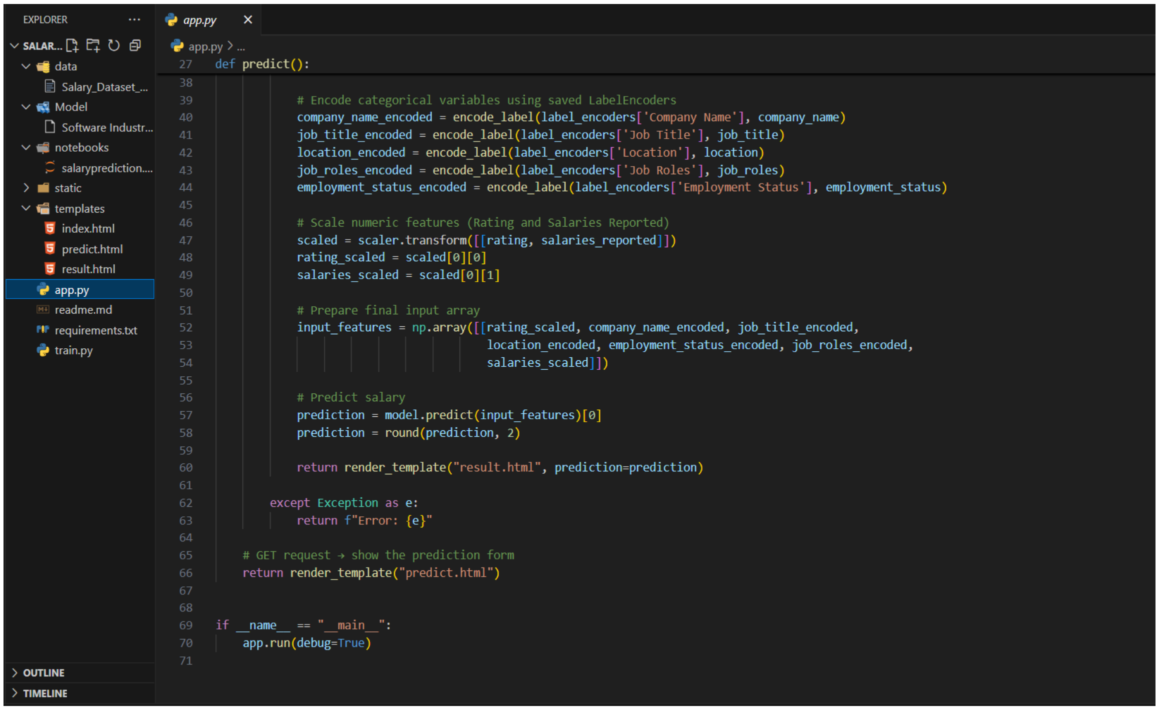


```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestRegressor
import pickle
import os

# Load dataset
df = pd.read_csv('data/Salary_Dataset_with_Extra_Features.csv')

# Fill missing values if any
df.fillna(method='ffill', inplace=True)

# Select features and target (now including Salary Reported)
features = ['Rating', 'Company Name', 'Job Title', 'Location',
            'Employment Status', 'Job Roles', 'Salaries Reported']
target = 'Salary'

X = df[features]
y = df[target]

# Encode categorical variables
label_encoders = {}
for col in ['Company Name', 'Job Title', 'Location', 'Employment Status', 'Job Roles']:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])
    label_encoders[col] = le

# Scale numeric features (Rating and Salary Reported)
scaler = StandardScaler()
X[['Rating', 'Salaries Reported']] = scaler.fit_transform(X[['Rating', 'Salaries Reported']])

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Train RandomForest model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Ensure Model directory exists
os.makedirs('Model', exist_ok=True)

# Save model, encoders, and scaler
with open('Model/Software Industry Salary Prediction.pkl', 'wb') as f:
    pickle.dump({
        'model': model,
        'label_encoders': label_encoders,
        'scaler': scaler
    }, f)

print("Model trained and saved successfully!")
```

Figure 2: Screenshot of app.py (Flask backend code)

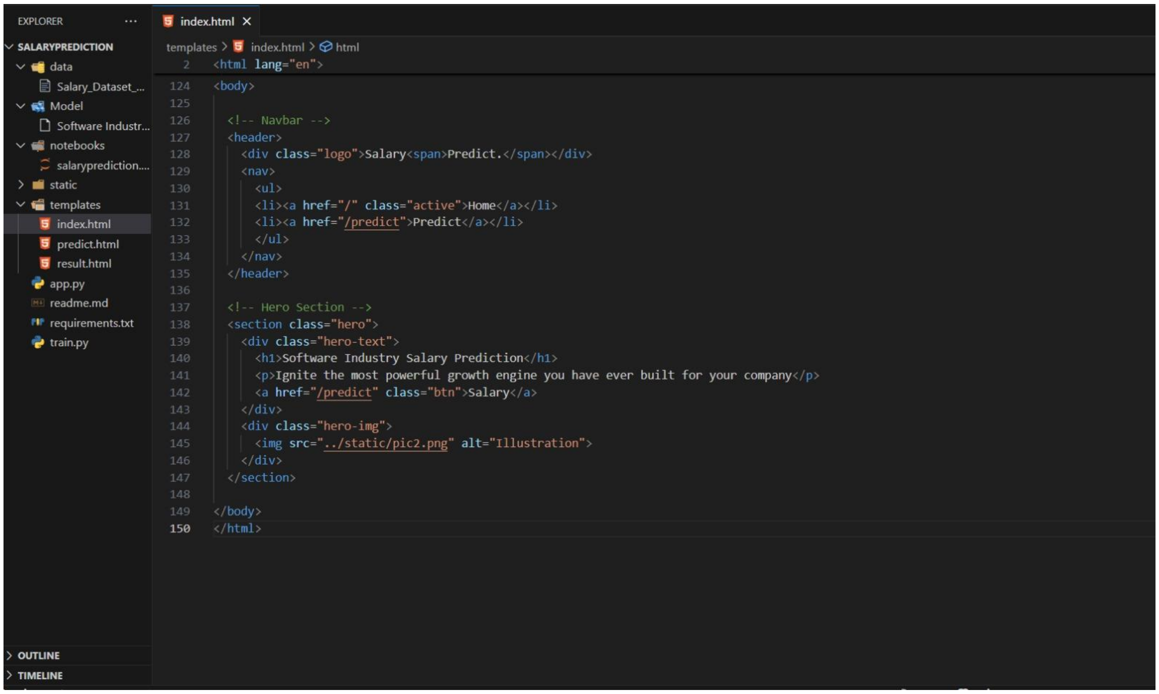


```python
from flask import Flask, render_template, request
import pickle
import numpy as np

app = Flask(__name__)

# Load the saved dictionary (model + label_encoders + scaler)
with open("Model/Software Industry Salary Prediction.pkl", "rb") as f:
    saved = pickle.load(f)

model = saved["model"]
label_encoders = saved["label_encoders"]
scaler = saved["scaler"]

def encode_label(le, value):
    if value in le.classes_:
        return le.transform([value])[0]
    else:
        # For unseen categories, return a default value (0)
        return 0

@app.route("/")
def index():
    return render_template("index.html")   # Landing page

@app.route("/predict", methods=["GET", "POST"])
def predict():
    if request.method == "POST":
        try:
            # Collect form data
            company_name = request.form["company_name"]
            job_title = request.form["job_title"]
            location = request.form["location"]
            job_roles = request.form["job_roles"]
            employment_status = request.form["employment_status"]
            salaries_reported = int(request.form["salaries_reported"])
            rating = float(request.form["rating"])
```



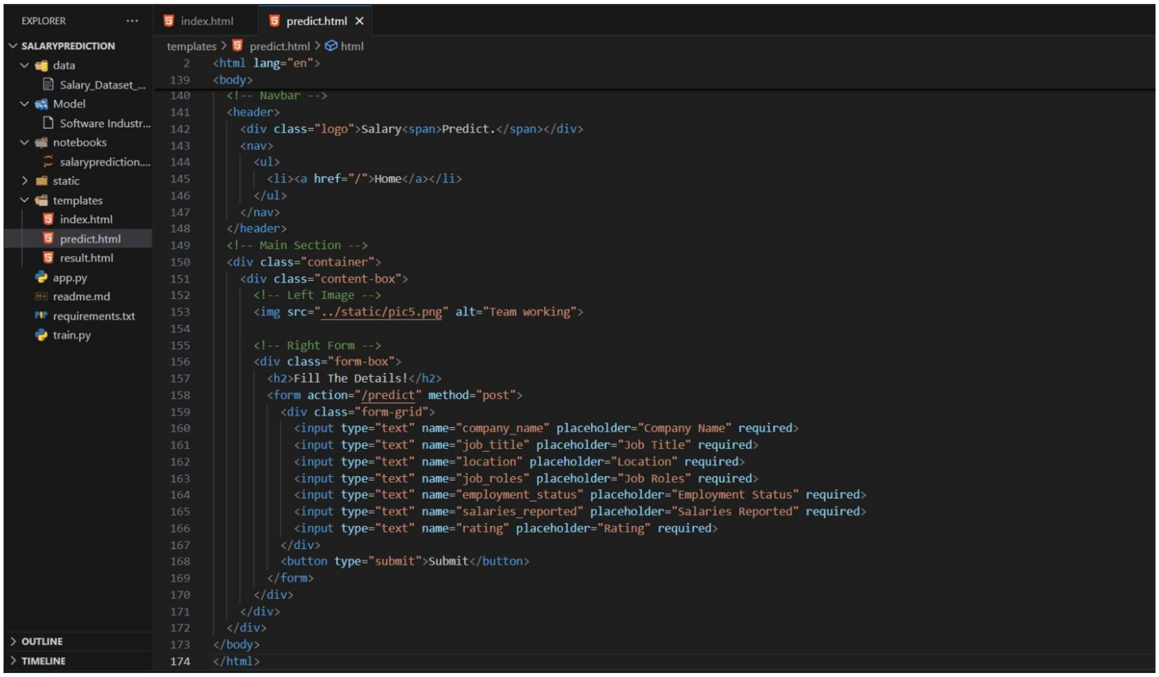
```python
def predict():

            # Encode categorical variables using saved LabelEncoders
            company_name_encoded = encode_label(label_encoders['Company Name'], company_name)
            job_title_encoded = encode_label(label_encoders['Job Title'], job_title)
            location_encoded = encode_label(label_encoders['Location'], location)
            job_roles_encoded = encode_label(label_encoders['Job Roles'], job_roles)
            employment_status_encoded = encode_label(label_encoders['Employment Status'], employment_status)

            # Scale numeric features (Rating and Salaries Reported)
            scaled = scaler.transform([[rating, salaries_reported]])
            rating_scaled = scaled[0][0]
            salaries_scaled = scaled[0][1]

            # Prepare final input array
            input_features = np.array([[rating_scaled, company_name_encoded, job_title_encoded,
                                       location_encoded, employment_status_encoded, job_roles_encoded,
                                       salaries_scaled]])

            # Predict salary
            prediction = model.predict(input_features)[0]
            prediction = round(prediction, 2)

            return render_template("result.html", prediction=prediction)

        except Exception as e:
            return f"Error: {e}"

    # GET request → show the prediction form
    return render_template("predict.html")


if __name__ == "__main__":
    app.run(debug=True)
```

Figure 3: Screenshot of index.html.



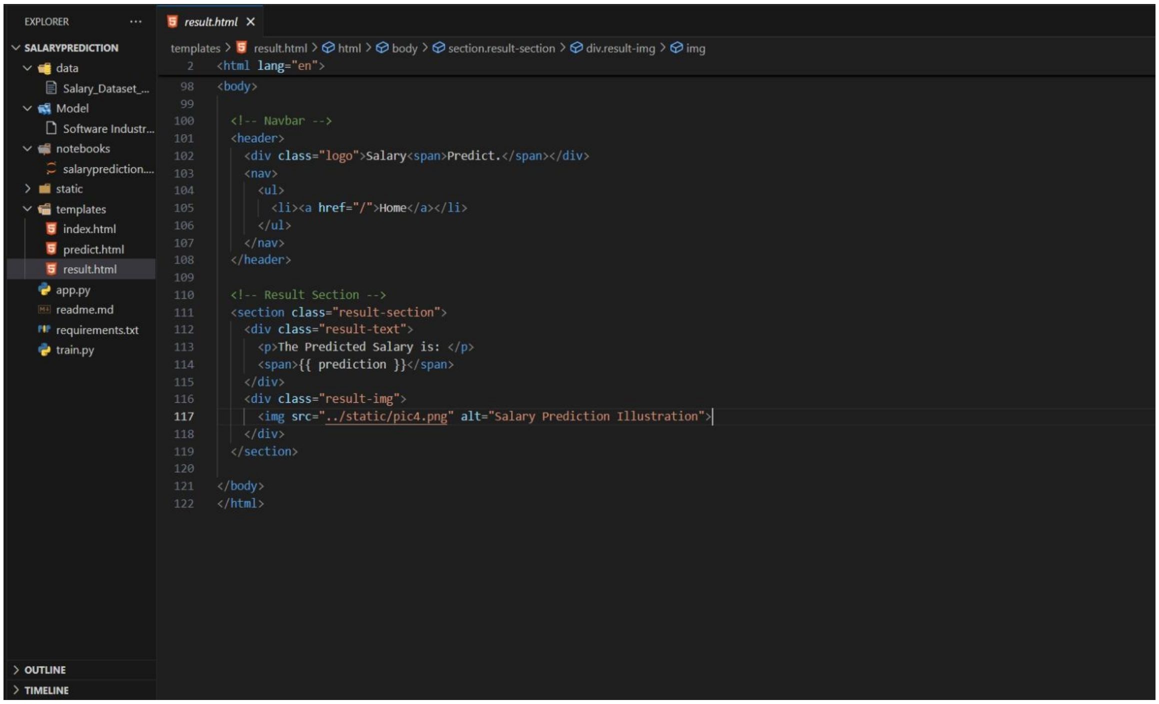
```html
<html lang="en">
<body>

  <!-- Navbar -->
  <header>
    <div class="logo">Salary<span>Predict.</span></div>
    <nav>
      <ul>
        <li><a href="/" class="active">Home</a></li>
        <li><a href="/predict">Predict</a></li>
      </ul>
    </nav>
  </header>

  <!-- Hero Section -->
  <section class="hero">
    <div class="hero-text">
      <h1>Software Industry Salary Prediction</h1>
      <p>Ignite the most powerful growth engine you have ever built for your company</p>
      <a href="/predict" class="btn">Salary</a>
    </div>
    <div class="hero-img">
      <img src="../static/pic2.png" alt="Illustration">
    </div>
  </section>

</body>
</html>
```

Figure 4: Screenshot of predict.html.



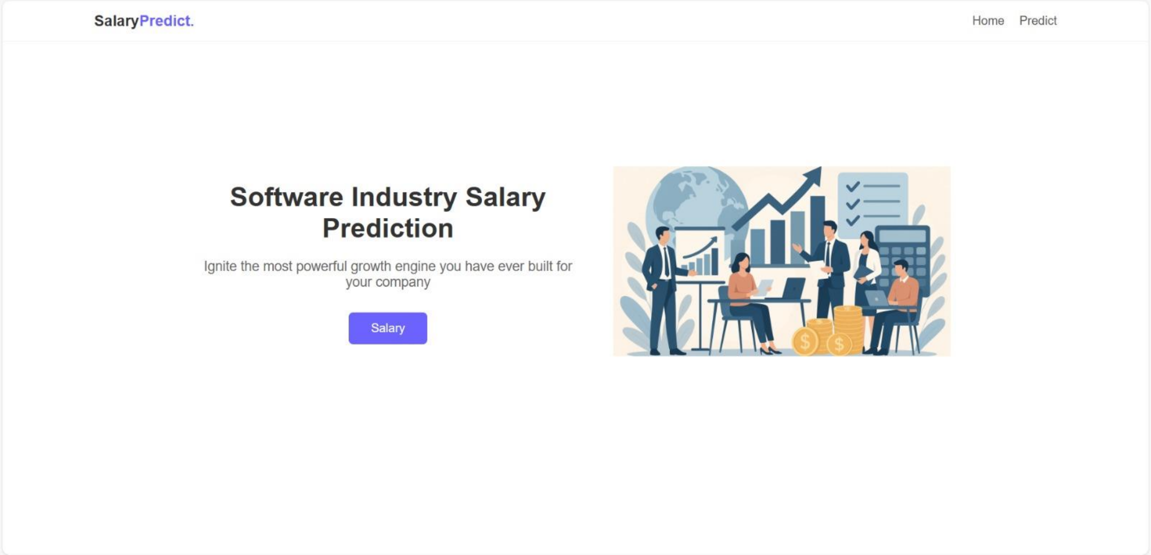
```html
<html lang="en">
<body>
  <!-- Navbar -->
  <header>
    <div class="logo">Salary<span>Predict.</span></div>
    <nav>
      <ul>
        <li><a href="/">Home</a></li>
      </ul>
    </nav>
  </header>
  <!-- Main Section -->
  <div class="container">
    <div class="content-box">
      <!-- Left Image -->
      <img src="../static/pic5.png" alt="Team working">

      <!-- Right Form -->
      <div class="form-box">
        <h2>Fill The Details!</h2>
        <form action="/predict" method="post">
          <div class="form-grid">
            <input type="text" name="company_name" placeholder="Company Name" required>
            <input type="text" name="job_title" placeholder="Job Title" required>
            <input type="text" name="location" placeholder="Location" required>
            <input type="text" name="job_roles" placeholder="Job Roles" required>
            <input type="text" name="employment_status" placeholder="Employment Status" required>
            <input type="text" name="salaries_reported" placeholder="Salaries Reported" required>
            <input type="text" name="rating" placeholder="Rating" required>
          </div>
          <button type="submit">Submit</button>
        </form>
      </div>
    </div>
  </div>
</body>
</html>
```

Figure 5: Screenshot of result.html.



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

  <!-- Navbar -->
  <header>
    <div class="logo">Salary<span>Predict.</span></div>
    <nav>
      <ul>
        <li><a href="/">Home</a></li>
      </ul>
    </nav>
  </header>

  <!-- Result Section -->
  <section class="result-section">
    <div class="result-text">
      <p>The Predicted Salary is: </p>
      <span>{{ prediction }}</span>
    </div>
    <div class="result-img">
      <img src="../static/pic4.png" alt="Salary Prediction Illustration">
    </div>
  </section>

</body>
</html>
```

Figure 6: Web Application – Index Page.



Software Industry Salary Prediction

Ignite the most powerful growth engine you have ever built for your company

Figure 7: Web Application – Predict Page.

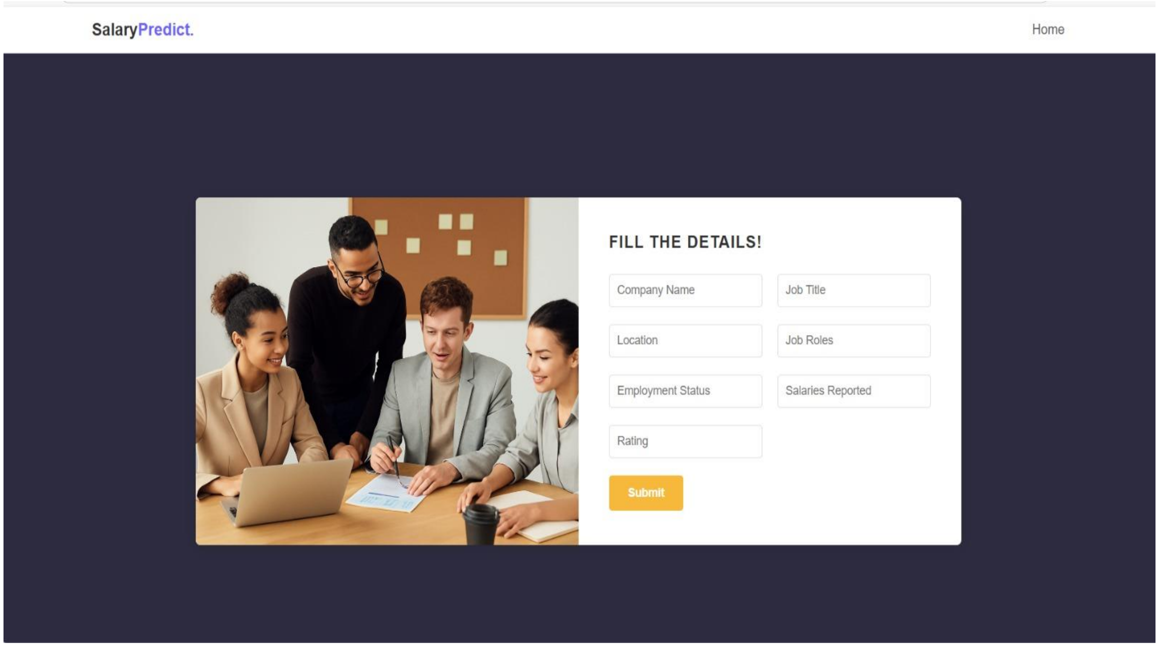
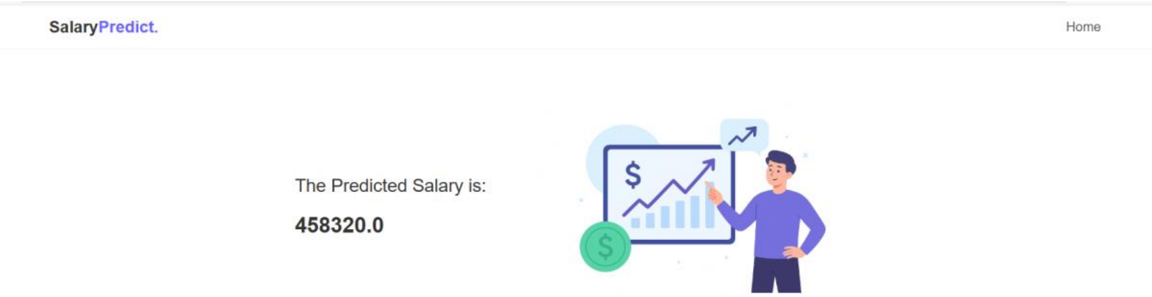


Figure 8: Web Application – Result Page.



The Predicted Salary is:

458320.0

GitHub Repository: https://github.com/shaikmanisha38/SalaryPrediction.git

Demo Video:
https://drive.google.com/file/d/1XVYv6Aa8gRUFIhFP4Z3to7w_SqF9t1yq/view?usp=drive_link

12. References

1. Python Documentation - https://docs.python.org/3/

2. Scikit-learn Library - https://scikit-learn.org/

3. Flask Framework - https://flask.palletsprojects.com/

4. Dataset Source - Internal Project Dataset