CAPSTONE PROJECT

Employee Salary Prediction

Presented By:
Vanshika Tomar-Indira Gandhi Delhi Technical University for Women-MAE

edunet foundation

OUTLINE

- Problem Statement
- System Development Approach
- Algorithm & Deployment
- Result
- Conclusion
- Future Scope
- References

PROBLEM STATEMENT

To predict an employee's expected salary based on input features (like education, experience, job title and age), and help HR professionals, job seekers, or recruiters make data-driven decisions.

REQUIRMENTS

System Requirements-
A system with at least 4GB RAM (8GB recommended), running Windows 10/macOS/Linux. Python 3.8+ and an IDE like VS Code or Jupyter Notebook are used for development.

Libraries Used-
The model is built using core Python libraries: pandas, numpy, scikit-learn, joblib, and streamlit for app deployment.

Model Requirements-
The Random Forest Regressor was used for salary prediction. It requires preprocessed input data and trained encoders for categorical features.

ALGORITHM

01 We loaded the employee salary dataset and performed preprocessing, including handling missing values and encoding categorical features like gender and department.

02 A Random Forest Regressor model was used to predict salaries. The data was split into training and testing sets, and the model was trained on the training data.

03 We evaluated the model using metrics like R^2 Score and Mean Absolute Error to ensure good prediction accuracy.

DEPLOYMENT

01 A Streamlit web application was created that allows users to input employee data or upload resumes. It displays the predicted salary instantly.

02 The trained model and preprocessing tools were saved using joblib, and the app was deployed on Streamlit Cloud using GitHub integration for version control.

RESULT

app.py | model_training.py | requirments.txt | Salary Data.csv

model_training.py > ...

```python
import pandas as pd
import joblib
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Load data
df = pd.read_csv("Salary Data.csv")
df.dropna(inplace=True)

# Label encode Gender and Education Level (ordinal)
label_encoders = {}
for col in ['Gender', 'Education Level']:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# One-hot encode Job Title
df = pd.get_dummies(df, columns=['Job Title'])

# Define features and target
X = df.drop(columns=['Salary'])
y = df['Salary']

# Save column order
feature_columns = X.columns.tolist()

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

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

# Save files
joblib.dump(model, "salary_model.pkl")
joblib.dump(label_encoders, "label_encoders.pkl")
joblib.dump(feature_columns, "feature_columns.pkl")

print("✅ Model trained and saved with one-hot encoding.")
```

app.py | model_training.py | requirments.txt | Salary Data.csv

app.py > ...

```python
# Load model and metadata
model = joblib.load("salary_model.pkl")
label_encoders = joblib.load("label_encoders.pkl")
feature_columns = joblib.load("feature_columns.pkl")  # All columns used during training

# Streamlit UI
st.set_page_config(page_title="Employee Salary Prediction", page_icon="💼", layout="centered")
st.title("💼 Employee Salary Prediction")

# Inputs
gender = st.selectbox("Gender", label_encoders["Gender"].classes_)
education = st.selectbox("Education Level", label_encoders["Education Level"].classes_)
job_title = st.selectbox("Job Title", sorted([col.replace("Job Title_", "") for col in feature_columns if col.startswith("Job Title_")]))
age = st.number_input("Age", min_value=18, max_value=70, value=25)
experience = st.number_input("Years of Experience", min_value=0, max_value=50, value=2)

# Encode inputs
try:
    gender_encoded = label_encoders["Gender"].transform([gender])[0]
    education_encoded = label_encoders["Education Level"].transform([education])[0]

    # Build initial input dictionary
    input_data = {
        "Gender": gender_encoded,
        "Education Level": education_encoded,
        "Age": age,
        "Years of Experience": experience
    }

    # One-hot encode job title
    for col in feature_columns:
        if col.startswith("Job Title_"):
            input_data[col] = 1 if col == f"Job Title_{job_title}" else 0

    # Convert to DataFrame with correct column order
    input_df = pd.DataFrame([input_data])
    input_df = input_df.reindex(columns=feature_columns, fill_value=0)

    # Predict
    if st.button("🔍 Predict Salary"):
        annual_salary = model.predict(input_df)[0]
        monthly_salary = annual_salary / 12

        st.success(f"💰 Predicted Monthly Salary: ${monthly_salary:,.2f}")
        st.info(f"📅 Predicted Annual Salary: ${annual_salary:,.2f}")
except Exception as e:
    st.error(f"Prediction failed: {e}")
```

model_training.py

app.py

edunet foundation

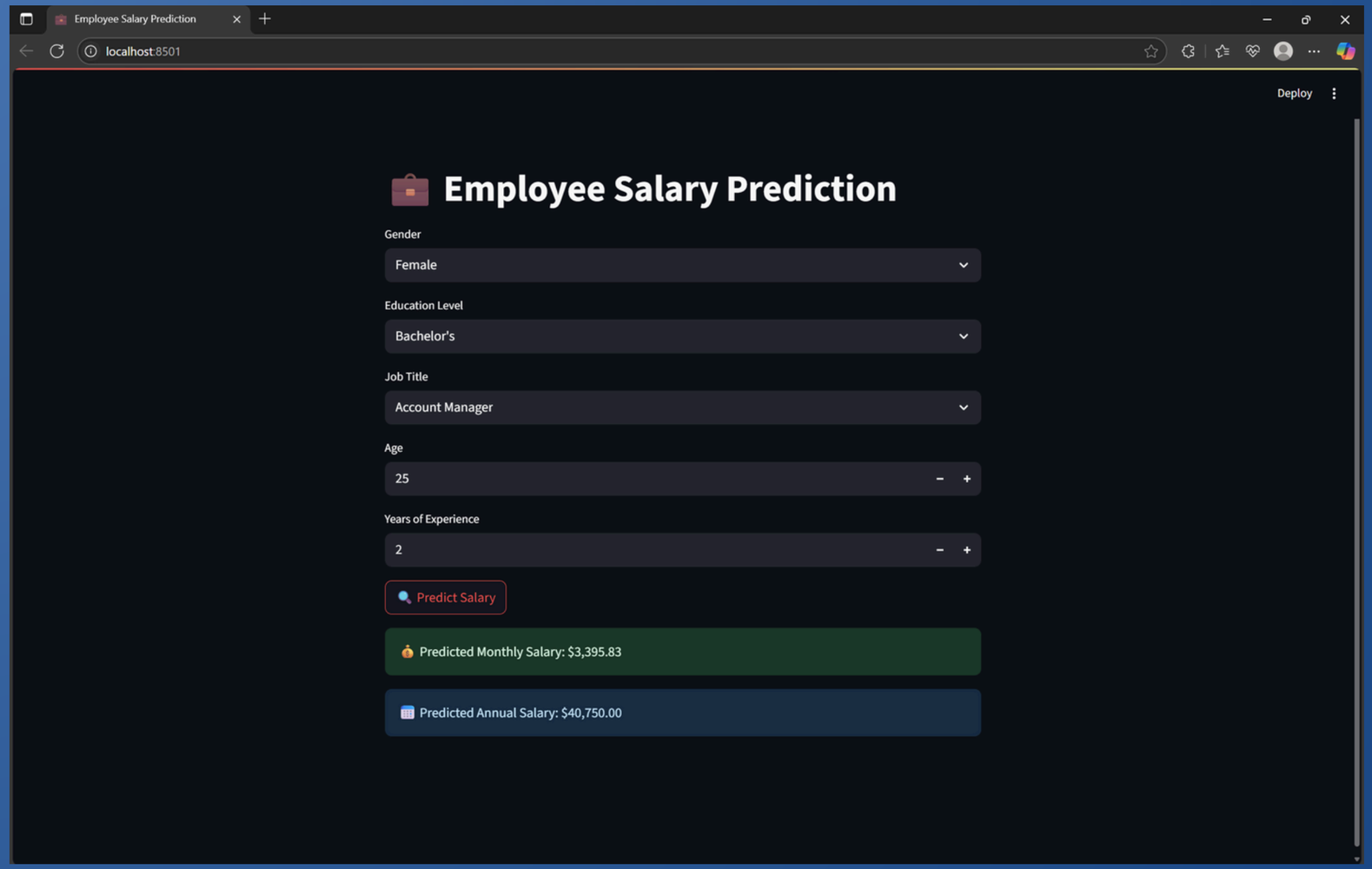
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\vansh\OneDrive\Documents\ibm intern\employee salary prediction> python model_training.py
✅ Model trained and saved with one-hot encoding.
PS C:\Users\vansh\OneDrive\Documents\ibm intern\employee salary prediction> streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.0.124:8501
```

Launching the streamlit app

1	Age	Gender	Education Level	Job Title	Years of Experience	Salary
2	32	Male	Bachelor's	Software Engineer	5	90000
3	28	Female	Master's	Data Analyst	3	65000
4	45	Male	PhD	Senior Manager	15	150000
5	36	Female	Bachelor's	Sales Associate	7	60000
6	52	Male	Master's	Director	20	200000
7	29	Male	Bachelor's	Marketing Analyst	2	55000
8	42	Female	Master's	Product Manager	12	120000
9	31	Male	Bachelor's	Sales Manager	4	80000
10	26	Female	Bachelor's	Marketing Coordinator	1	45000

1st ten rows of the Dataset used

edunet foundation



Output

Github Link: https://github.com/Vanshika1664/employee-salary-prediction

edunet foundation

CONCLUSION

01

This project successfully developed a machine learning model using the Random Forest algorithm to predict employee salaries based on various features such as education level, total experience, job role, and work location.

02

A user-friendly web application was built using Streamlit, allowing users to either manually input details or upload resumes for automated data extraction, enhancing accessibility and convenience.

03

The model provides consistent and logical salary predictions, making it a valuable tool for both employees and employers to estimate fair compensation and support HR decision-making processes.

edunet foundation

FUTURE SCOPE

MODEL ENHANCEMENT

Implement more advanced machine learning algorithms like XGBoost or neural networks to further improve the model's accuracy and handle more complex patterns.

DATA EXPANSION

Extend the dataset by incorporating data from diverse industries, global regions, and job levels to make the model more robust and generalizable.

AUTOMATED INPUT SYSTEMS

Add functionality to extract candidate information automatically using resumes or LinkedIn profiles for a smoother user experience.

unet
foundation

REFRENCES

Salary Data.csv Dataset – **Kaggle**.

GitHub Repositories and Stack Overflow for debugging and implementation references.

Kaggle & Medium articles for best practices in employee attrition/salary prediction modeling.

AICTE – IBM SkillsBuild AI/ML Virtual Internship Program by Edunet Foundation.

edunet foundation

THANK YOU

edunet
foundation