Bangalore Institute of Technology
K.R Road, V.V Pura, Bengaluru-560004
Department of Computer Science & Engineering

"DevPay.pro : Anticipating Software Developer Earning in India"

Presentation by
AARYAN GAUTAM - 1BI21CS001
ADARSH KUMAR - 1BI21CS006
AUM JALINDAR JADHAV- 1BI21CS029

Under the Guidance of
Prof. J. Girija
Professor & HOD
Department of CSE, BIT

Agenda

- Introduction
- Existing Systems
- Problem Statement
- Objectives
- Tools and Technologies
- Architectural Diagram
- Modules Description
- Implementation
- Results
- Applications
- Conclusion and Future scope

INTRODUCTION

- **DevPay Pro**: A transformative solution for the Indian software development community.

- **Key Features**:
  - Unprecedented insights into software developer earnings.
  - Intuitive interface with robust backend architecture.
  - Personalized salary projections.
  - Scalability and affordability.

EXISTING SYSTEMS

- **Challenges with existing systems**:
  1) Fragmented data sources.
  2) Inaccurate projections.
  3) High subscription costs.
  4) Using in-efficient models for prediction.

- **Need for a Comprehensive Solution**: Addressing these issues with DevPay Pro.

PROBLEM STATEMENT

In the rapidly evolving tech industry, understanding and predicting developer salaries is crucial for both employers and employees. Traditional salary surveys often fail to capture the dynamic nature of the market, leading to outdated or generalized insights. DevPay.Pro aims to address this gap by providing a robust, data-driven platform to predict developer salaries based on various attributes such as years of coding experience, educational background, programming languages known, and geographical location.

OBJECTIVES

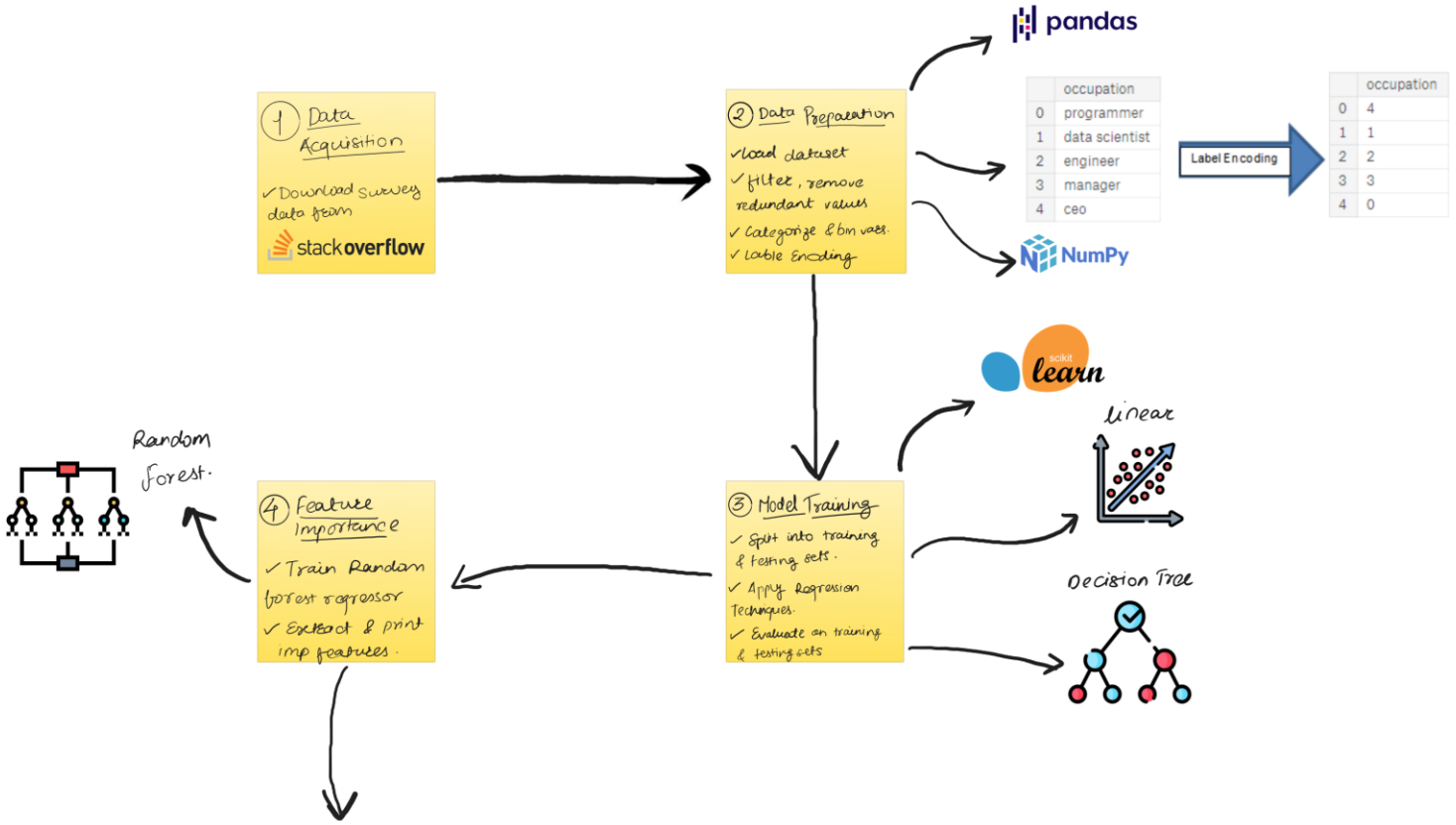
Develop a predictive model that accurately forecasts developer salaries using machine learning techniques. The model should incorporate multiple features, including but not limited to:

- Years of coding experience
- Educational degree
- Programming languages known
- Country/Region
- Industry sector

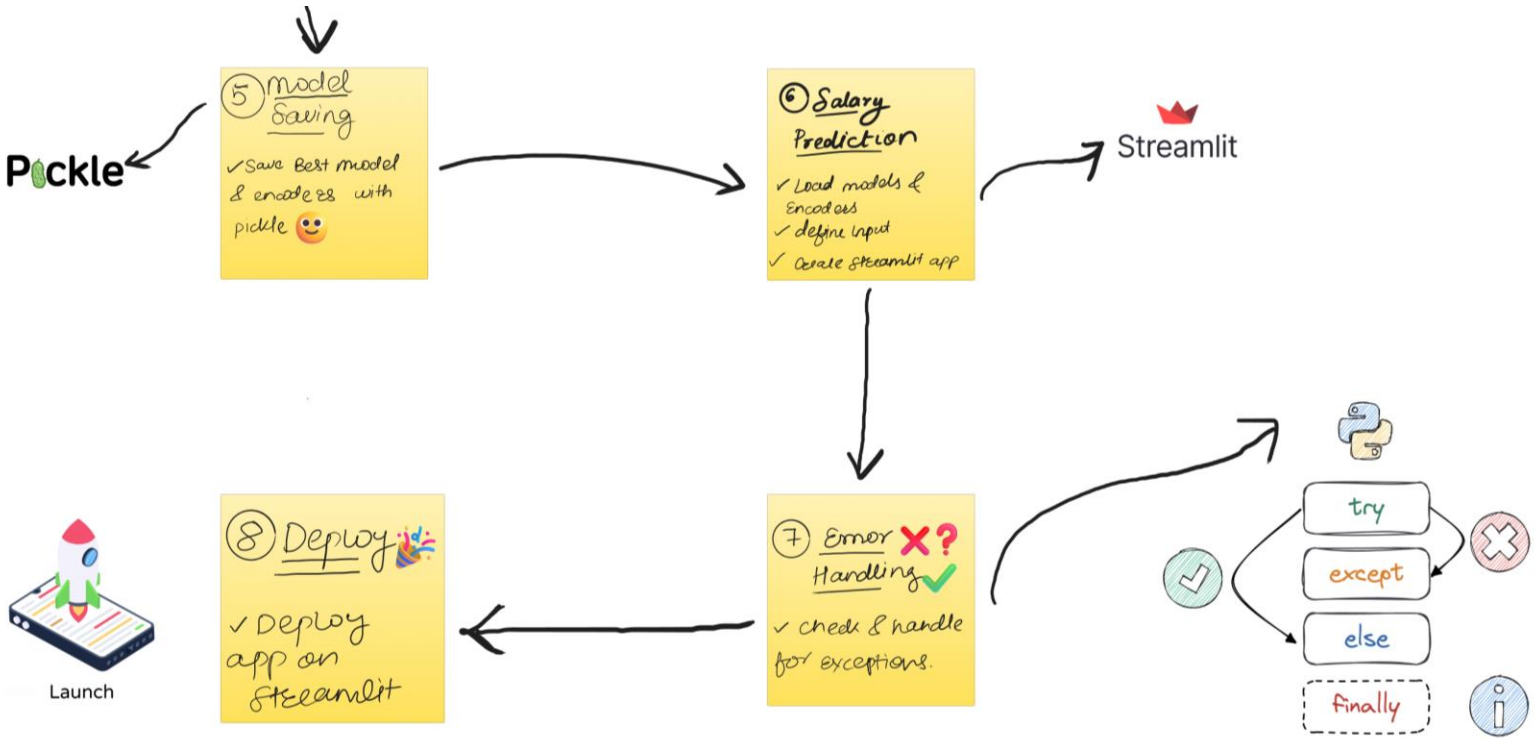
TOOLS AND TECHNOLOGIES

- **Frontend**:
  - StreamLit
  - HTML/CSS
- **Backend:**
  - Python - StreamLit
- **Data Analytics**:
  - Sci-kit Learn
  - Numpy
  - Pandas
  - Matplotlib
  - Machine Learning Algorithms

ARCHITECTURAL DIAGRAM



ARCHITECTURAL DIAGRAM



MODULES DESCRIPTION

- **Data Input and Validation:**
- Interface for users to input their details such as years of experience, education level, programming languages known, etc. Includes validation to ensure data accuracy.

- **Data Preprocessing:**
- Backend module to preprocess user-inputted data, handling missing values, encoding categorical variables, and scaling numerical features.

MODULES DESCRIPTION

- **Machine Learning Model Training:**
- Module for training machine learning models using historical salary data and the user-provided features. Includes selecting the appropriate model (e.g., regression, ensemble methods) and optimizing hyperparameters.

- **Prediction Engine:**
- Component that generates salary predictions based on the trained models and user inputs. Provides explanations or insights into how each feature affects the predicted salary.

MODULES DESCRIPTION

- **Visualization and Reporting:**
- Module to create interactive visualizations (e.g., salary distribution charts, trend graphs) that help users explore and understand salary insights based on different criteria (e.g., location, job role).

- **User Interface (UI) Design:**
- Development of a user-friendly interface using frameworks like Streamlit or Flask to facilitate easy navigation and interaction with the application.

- **Deployment and Hosting:**
- Implementation of the application on a web server, ensuring scalability and reliability. Integration with cloud services (e.g., AWS, Azure) for efficient hosting and management.

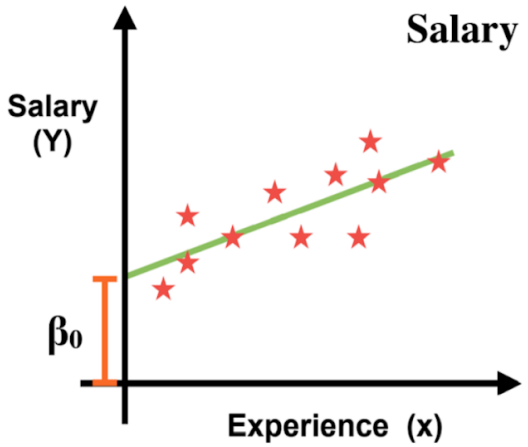
MODULES DESCRIPTION

- **Documentation and Help Section:**
- Creation of comprehensive documentation explaining how to use the application, interpret results, and troubleshoot common issues. Includes a help section or FAQs

IMPLEMENTATION

Algorithms (or) Code Snippet (or) Pseudocode

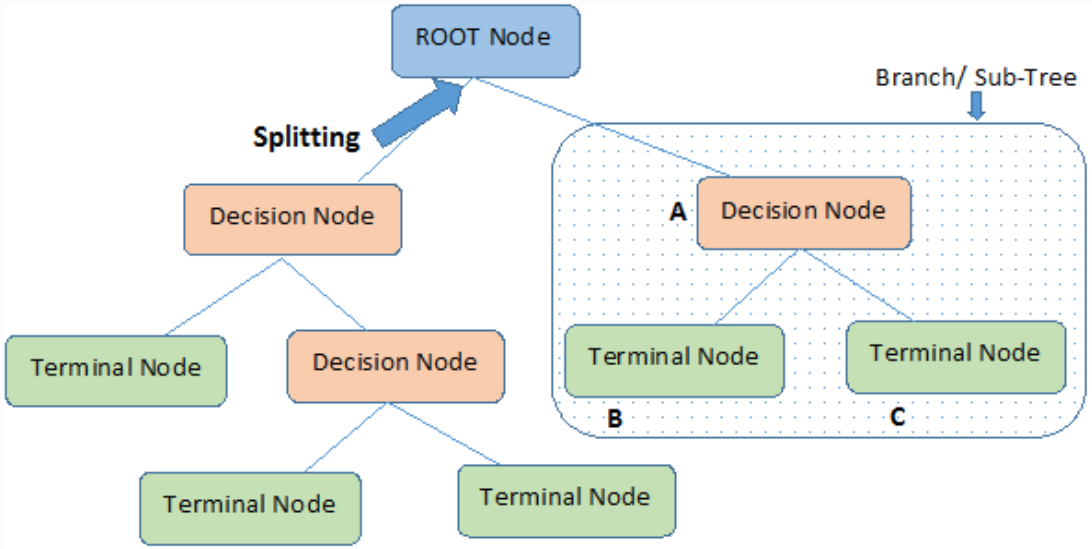
Linear Regression



$$\text{Salary} = \beta_0 + \beta_1 * \text{Experience}$$

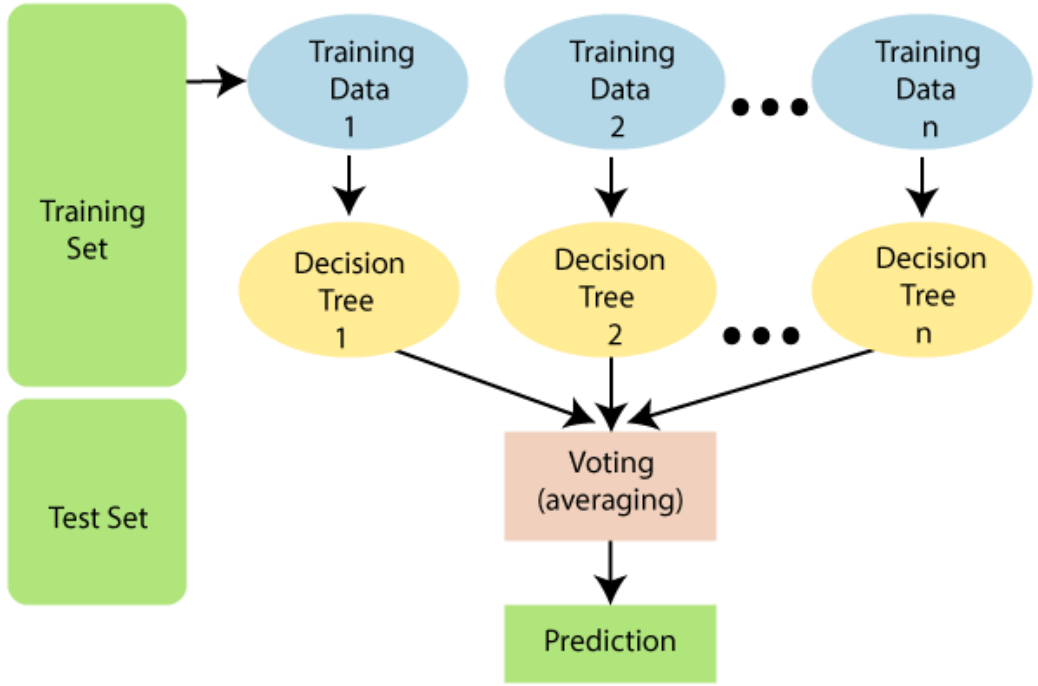
$$\beta_0 = \$\ 30K$$

Decision Tree



IMPLEMENTATION

Algorithms (or) Code Snippet (or) Pseudocode



Random Forest Algorithm

IMPLEMENTATION

Algorithms (or) Code Snippet (or) Pseudocode

Developer's Salary Prediction.ipynb | app.py 3 | requirements.txt | home_page.py | social_page

app.py > ...

```python
import streamlit as st
import streamlit_option_menu
from streamlit_option_menu import option_menu
from predict_page import show_predict_page
from explore_page import show_explore_page
from home_page import show_home_page
from social_page import show_socials_page
logo_path = "menulogo.png"

with st.sidebar:
    st.image(logo_path, use_column_width=True)
    st.title("Main Menu")
    selected2 = option_menu(None, ["Home", "Predict", "Explore", 'Socials'],
    icons=['house', 'cloud-upload', "search", 'gear'],
    menu_icon="cast", default_index=1, orientation="vertical")


if selected2 == "Predict":
    show_predict_page()
elif selected2=="Explore":
    show_explore_page()
elif selected2=="Home":
    show_home_page()
elif selected2=="Socials":
    show_socials_page()
```

Developer's Salary Prediction.ipynb | app.py 3 | requirements.txt | home_page.py 2 | social_pa

explore_page.py > show_explore_page

```python
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt

# Create a function to compile the countries with smaller values into "Others" categor
def compile_countries(categories, cutoff):
    categorical_map = {}
    for i in range(len(categories)):
        if categories.values[i] >= cutoff:
            categorical_map[categories.index[i]] = categories.index[i]
        else:
            categorical_map[categories.index[i]] = "Others"
    return categorical_map

# Create a function to standardize and clean the coding experience column
# Convert Coding Experience Column into float data type
def clean_yearsCodePro(x):
    if x == 'More than 50 years':
        return 50
    if x == 'Less than 1 year':
        return 0.5
    return float(x)

# Clean education column into a few categories
def clean_edLevel(x):
    if "Bachelor's degree" in x:
        return "Bachelor's degree"
    if "Master's degree" in x:
        return "Master's degree"
    if "Professional degree" in x or "Other doctoral" in x:
        return "Postgraduate"
    return "Less than a Bachelors"
```

IMPLEMENTATION

Algorithms (or) Code Snippet (or) Pseudocode

Developer's Salary Prediction.ipynb | app.py 3 | requirements.txt | home_page.py 2 | social_pa

predict_page.py > load_model

```python
import streamlit as st
import pickle
import numpy as np


@st.cache_data
def load_model():
    with open('saved_steps.pkl', 'rb') as file:
        data = pickle.load(file)
    return data

data = load_model()

regressor = data['model']
le_country = data['le_country']
le_edLevel = data['le_edLevel']
le_remoteWork = data['le_remoteWork']

def show_predict_page():
    st.title("Developers Salary Prediction")

    st.write("""#### Please provide some information to us in order to predict the sal

    countries = ("United States of America",
                 "Germany",
                 "United Kingdom of Great Britain and Northern Ireland",
                 "India",
                 "Canada",
                 "Brazil",
                 "France",
                 "Spain",
                 "Netherlands",
                 "Italy",
```

Developer's Salary Prediction.ipynb | app.py 3 | requirements.txt | home_page.py 2 | social_pa

explore_page.py > show_explore_page

```python
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt

# Create a function to compile the countries with smaller values into "Others" categor
def compile_countries(categories, cutoff):
    categorical_map = {}
    for i in range(len(categories)):
        if categories.values[i] >= cutoff:
            categorical_map[categories.index[i]] = categories.index[i]
        else:
            categorical_map[categories.index[i]] = "Others"
    return categorical_map

# Create a function to standardize and clean the coding experience column
# Convert Coding Experience Column into float data type
def clean_yearsCodePro(x):
    if x == 'More than 50 years':
        return 50
    if x == 'Less than 1 year':
        return 0.5
    return float(x)

# Clean education column into a few categories
def clean_edLevel(x):
    if "Bachelor's degree" in x:
        return "Bachelor's degree"
    if "Master's degree" in x:
        return "Master's degree"
    if "Professional degree" in x or "Other doctoral" in x:
        return "Postgraduate"
    return "Less than a Bachelors"
```

APPLICATIONS

Salary Negotiation Tool for Developers:

- Developers can use DevPay.Pro to understand their market value based on their skills, experience, and location. This empowers them to negotiate better salaries and benefits with current or prospective employers.

Recruitment and Hiring Insights for Employers:

- Employers and HR professionals can utilize DevPay.Pro to establish competitive and fair salary offers for potential hires. This helps attract top talent and ensures compensation packages are aligned with industry standards.

APPLICATIONS

Career Planning and Development:

- Developers can plan their career paths more effectively by exploring how different factors (e.g., learning new programming languages, gaining additional certifications) impact salary potential.

Market Analysis for Industry Professionals:

- Industry analysts and researchers can leverage DevPay.Pro to analyze salary trends and patterns across different regions, job roles, and skill sets. This information can be used to publish reports and provide strategic insights.

Freelancer Rate Setting:

- Freelancers and independent contractors can use the platform to set competitive hourly rates based on their expertise and the current market demand for their skills.

CONCLUSION AND FUTURE WORK

- **Current Impact:**
  - DevPay Pro empowers developers with essential salary insights.
- **Future Enhancements:**
  - Expanding data sources and analytics capabilities.
  - Incorporating more personalized career planning tools.
- **Long-Term Vision**: Continuous improvement to support the evolving needs of the tech industry.

Thank-You