**Dashboard Summary**

**Page 1: Executive Overview**

* **Purpose**: Gives HR a high-level view of employee distribution, average salaries, top roles, and satisfaction metrics.
* **Key Elements**:
  + **KPI Cards**: Avg Salary, Avg Experience, Job Roles, Total Employees, Job Satisfaction.
  + **Visuals**:
    - Bar Chart: Top 5 Departments by Avg Salary.
    - Pie Chart: Gender Distribution.
    - Line Chart: Salary by Experience.
    - Bar Chart: Top 5 Job Roles by Count.

**Page 2: Skill Demand Analysis**

* **Purpose**: Help HR understand which skills are most in-demand and how they map to roles.
* **Visuals**:
  + Skill Treemap
  + Required Skills by Job Role (Bar Chart)
  + Department vs Skill Demand (Matrix)

**Page 3: Salary Insights**

* **Purpose**: Provide deep insights into compensation trends.
* **Visuals**:
  + Salary Range Summary
  + Avg Monthly Income by Job Role
  + Salary by Experience
  + % Salary Hike by Department
  + Top 5 Departments by Avg Salary

**Page 4: Visa & Job Status**

* **Purpose**: Identify visa-based hiring opportunities and job availability.
* **Visuals**:
  + Visa Status by Role
  + Avg Salary by Visa Status
  + Job Status by Department
  + Visa Count by Department

**Page 5: Settings Page**

**Purpose**: Presents the filters that apply to all the pages in the dashboard. Helps to view the dashboard In different perspectives.

**Predictive Salary Model (Python Integration)**

The salary prediction model implemented in this Power BI dashboard uses a supervised machine learning approach to estimate monthly salaries for final-round candidates based on their experience and skill sets. The model is built using Python, integrated within Power BI through a Python visual, and leverages the RandomForestRegressor from the scikit-learn library for its robustness and ability to model complex, non-linear relationships. The process begins by training the model on historical employee data, where the RequiredSkills column, initially stored as comma-separated strings, is transformed into binary features using MultiLabelBinarizer. This one-hot encoding process ensures that each distinct skill becomes an individual input feature for the model. The TotalWorkingYears is also included as a numerical input, while the target variable is MonthlyIncome. Once trained, the model is used to predict salaries for new candidates whose data is pulled from a secondary table within Power BI. These candidates have attributes like predicted JobRole, TotalWorkingYears, and a list of required skills, which are preprocessed in the same way as the training data. After cleaning and encoding, the model makes predictions which are stored in a new column, PredictedSalary. The results are then exported to a CSV file and visualized using a bar chart that maps CandidateID to their predicted income. This integration of machine learning with real-time business intelligence tools like Power BI empowers HR teams to make data-driven compensation decisions and proactively evaluate candidate fit based on technical and experiential alignment.

import pandas as pd

from sklearn.ensemble import RandomForestRegressor

from sklearn.preprocessing import MultiLabelBinarizer

import matplotlib.pyplot as plt

# --- Step 1: Train model with historical data ---

df\_train = dataset.copy()

df\_train['RequiredSkills'] = df\_train['RequiredSkills'].fillna("").apply(lambda x: x.split(', '))

mlb = MultiLabelBinarizer()

skills\_encoded\_train = mlb.fit\_transform(df\_train['RequiredSkills'])

X\_train = pd.concat([

df\_train[['TotalWorkingYears']].reset\_index(drop=True),

pd.DataFrame(skills\_encoded\_train, columns=mlb.classes\_)

], axis=1)

y\_train = df\_train['MonthlyIncome']

model = RandomForestRegressor()

model.fit(X\_train, y\_train)

# --- Step 2: Load final-round candidates ---

candidates = dataset[dataset['CandidateID\_predict'].notnull()].copy()

candidates = pd.DataFrame({

'CandidateID': candidates['CandidateID\_predict'],

'JobRole': candidates['JobRole\_predict'],

'TotalWorkingYears': candidates['TotalWorkingYears\_predict'],

'RequiredSkills': candidates['RequiredSkills\_predict'].apply(lambda x: [x] if pd.notnull(x) else [])

})

candidates.dropna(subset=['CandidateID', 'JobRole', 'TotalWorkingYears', 'RequiredSkills'], inplace=True)

# --- Step 3: Encode and Predict ---

skills\_encoded\_candidates = mlb.transform(candidates['RequiredSkills'])

X\_test = pd.concat([

candidates[['TotalWorkingYears']].reset\_index(drop=True),

pd.DataFrame(skills\_encoded\_candidates, columns=mlb.classes\_)

], axis=1)

candidates['PredictedSalary'] = model.predict(X\_test)

# --- Step 4: Save as CSV ---

candidates = candidates.drop\_duplicates(subset='CandidateID')

try:

candidates.to\_csv(r'C:\Users\Public\predicted\_salaries.csv', index=False)

except:

print("Could not save to local path. Likely running in Power BI sandbox.")

# --- Step 5: Show bar chart ---

candidates['CandidateID'] = candidates['CandidateID'].astype(str)

plt.figure(figsize=(8, 5))

plt.bar(candidates['CandidateID'], candidates['PredictedSalary'], color='#4B425A')

plt.xlabel('Candidate ID')

plt.ylabel('Predicted Salary')

plt.title('Predicted Salary for Final-Round Candidates')

plt.xticks(rotation=30)

plt.grid(axis='y')

plt.tight\_layout()

plt.show()

A graph of a salary

AI-generated content may be incorrect.