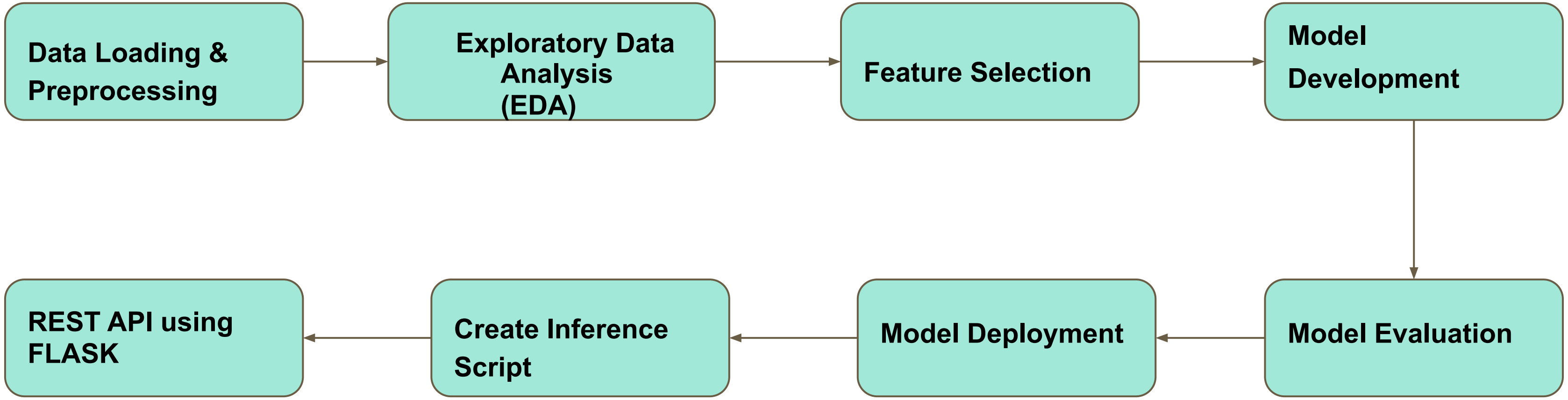
Employee Income Prediction

Prepared by : Rachana Baldania

Objective and Architecture

Objective: Predict whether an employee earns more than $50K per year.

Problem Type: Supervised Classification as labels are discrete and binary.



1. Importing Libraries

- **Data Analysis and Visualization:**
  - Pandas
  - Matplotlib
  - Seaborn
- **Machine Learning:**
  - Scikit-learn
  - XGBoost
- **Additional Libraries:**
  - Imbalanced-learn
  - sklearn-pandas

2. Data Loading & Overview

- **Dataset Information:**
  - Number of Rows: 32,561
  - Number of Columns: 15
- **Attributes**: List the dataset attributes (age, workclass, education, etc.).
- **Target Variable**: Income (>$50K or <=$50K per year)
- **Data Types:** Numerical and Categorical
- **Statistical Analysis of Data:** mean, median, mode etc

3. Data Preprocessing and Exploratory Data Analysis (EDA)

Data Cleaning: Handling missing values, duplicates, and outliers.

- Example:
  - Remove whitespace from column names and elements
  - Convert target column to numerical classes (0 and 1)
  - Handle missing values ('?' replaced with NaN)
  - Impute missing values using mode for categorical columns
  - Drop redundant columns (e.g., `education-num`)
  - Identifying Outliers

Data Visualizations: Show histograms, bar charts, and scatter plots to illustrate the data distribution and relationships.

- Example:
  - Distribution of Income
  - Age Distribution
  - Workclass Analysis
  - Education and Education Number
  - Gender and Marital Status
  - Relationship, Race, and Native Country
  - Occupation and Hours per Week

Data Transformation: Encoding categorical variables, normalizing continuous variables.

- Example:
- Converted categorical columns to dummy variables
- Encoded categorical variables using techniques like One-Hot Encoding or Label Encoding.Transformed categorical columns using Label Encoding.
- Normalize or standardize continuous features (e.g., `age`, `fnlwgt`, `capital-gain`, `capital-loss`, `hours-per-week`).StandardScaler applied to numerical features

Interpretation of EDA

- **Income:**The distribution of the target column, I find that the people with less than or equals to 50K annual income are 75.92% and the no. of people making more than 50K is 24.08%, so it's clear that the dataset is unbalanced.
- **Age:** there are relatively less young people who have an annual income is more than 50K.
- **Workclass:** who have an annual income is more than 50K or less than 50K are work in private
- **Education and Education no:** education & education number column are just the same.so I am dropped education number column.And In this dataset Most number of people are high school graduates with 9 to 10 years of education.
- **Gender:**more percentage of male candidates are earn more than 50K.
- **Marital Status:** Most of the people whose annual income is more than 50K, their marital status is married.
- **Relationship:** Most of the people whose annual income is more than 50K, are husband and not in family so we can consider any one of them in further experiment Marital status or Relationship
- **race:**In this dataset majority of information about white race is more while all other races are lesser.
- **Country:**91.38% peoples are from United-states and only 8.62% people belong from other countries.so divided cols in US and Others category
- **Occupation:** In occupation column the values are well distributed in all categories. We can ignore this in further experiment
- **Hours per week:** most of the people work 30 to 40 hours per week

4.Feature Selection:

Use techniques like correlation analysis, mutual information, or feature importance from models to select significant features.

Dropped redundant columns.

Correlation analysis to identify significant features.

Selected final features for modeling Dropped low-correlation features.

Handling Data Imbalance

To handle uneven dataset target class distribution, >=50 has less sample so need to oversample it

Used RandomOverSampler to balance the dataset.

Visualized class distribution before and after sampling.

```
Before sampling :

income

0    24720

1     7841
```

Results

Key features influencing income prediction:

```
marital-status    0.434944

relationship      0.250918

age               0.234037

hours-per-week    0.229689

capital-gain      0.223329

sex               0.215980

capital-loss      0.150526
```

```
After sampling :

income

0    24720

1    24720
```

5.Model Development

- **Training and Validation**: Explain the process (train-test split, cross-validation).
- **Modes Selection**: Logistic Regression,Decision Tree Classifier,Random Forest Classifier,Support Vector Classifier,Gradient Boosting Classifier,XGBoost Classifier
- **Model Performance Comparison**: Show a comparison of model performances.Compared models' accuracy and F1 score and out of all model XGBoost achieved highest accuracy and F1 score which is around 86% for both train and test data. So further optimize model through hyperparameter tuning
   a. **Hyperparameter Optimization**
   - **Approach:** RandomizedSearchCV
   - **Best Parameters Identified:**
      - Learning rate
      - Max depth
      - Min child weight
      - Gamma
      - Colsample bytree

6.Model Evaluation

- **Evaluation Metrics based on Tuned XGBoost Classifier**: Confusion matrix :Accuracy, precision, recall, F1-score and ROC-AUC.
- Save the model pickle file for Deployment.
- Result Interpretation: So as I try different Classification Algorithms and found that XGBoost Classifier is Performed the best with an F1 score of 0.88 AUC Score of 0.95 and Accuracy score of 0.87. I also Performed hyperparameter tuning on XGBoost classifier and get the best tuned parameter.Used multiple Performance metrics to ensure that the model is performing correctly and is not overfitting on the data.

- **Insights**: Key findings from the model.
   a. Example: age, capital_gain, capital_loss, hours_per_week, education, sex, marital_status, country,race,relationship,occupation are strong indicators of higher income.
   b. The most number of people are young, white, male, high school graduates with 9 to 10 years of education, work 40 hours per week and belong from United-States.
   c. While doing exploratory analysis, I found out that the most no. of people are young. But relatively less young people who have an annual income is more than $50K.
- **Business Implications:** How the insights can help in decision-making.
- Example: Targeted training programs for employees in lower-income brackets.

Future Work: Potential improvements and future research directions.

   a. Recap of the problem, methodology, and key findings.
   b. Example: Incorporating more features, using deep learning models.
   c. Can remove race,relationship

7.Model Deployment

- **Steps to Deploy the Model:**
   - Save the trained model using joblib or pickle
   - Create Inference script
   - Develop a REST API using Flask or FastAPI
   - Containerize the application using Docker
   - Deploy on cloud platforms like AWS, Azure, or GCP
   - Monitor model performance and update as needed
- **I have create Deployment folder in provided jupyter notebook where i have created inference script and developed REST API using Flask.**

System Architecture Design

1. Data Ingestion

- **Data Sources:** The data can be collected from various sources such as databases, CSV files, or APIs.
- **Data Ingestion Service:** A service that collects and ingests data into a central repository. Tools like Apache Kafka or AWS Kinesis can be used for real-time data ingestion, while batch data can be handled using ETL tools like Apache NiFi or Airflow.

2. Data Storage

- **Data Lake:** Store raw data in a scalable and cost-effective storage solution like Amazon S3, Google Cloud Storage, or Azure Blob Storage.
- **Data Warehouse:** For structured and cleaned data, use a data warehouse like Amazon Redshift, Google BigQuery, or Snowflake.

3. Data Preprocessing

- **ETL Pipelines:** Use ETL tools to clean, transform, and load data. Apache Airflow can manage workflows and schedule tasks.
- **Preprocessing Scripts:** Python scripts using libraries like pandas, scikit-learn, and Dask for data cleaning, encoding, and scaling.
- **Feature Store:** A dedicated feature store (e.g., Feast) to store and manage features.

4. Model Training

- **Training Environment:** Use Jupyter notebooks or script-based training on local machines for initial development, and scalable cloud services like AWS SageMaker, Google AI Platform, or Azure ML for large-scale training.
- **Model Versioning:** Tools like MLflow or DVC for tracking experiments, models, and datasets.

5. Model Deployment

- **Model Serving:** Deploy models using a scalable serving infrastructure like TensorFlow Serving, TorchServe, or custom APIs using Flask/FastAPI.
- **Containerization:** Use Docker to containerize the model serving infrastructure.
- **Orchestration:** Kubernetes to manage and scale containers.

6. Prediction Service

- **API Gateway:** AWS API Gateway or similar to expose the prediction service to external applications.
- **Load Balancer:** Distribute incoming requests to multiple instances of the prediction service.

7. Monitoring and Logging

- **Monitoring Tools:** Prometheus and Grafana for monitoring performance metrics of the model and infrastructure.
- **Logging:** ELK stack (Elasticsearch, Logstash, Kibana) or cloud-native solutions like AWS CloudWatch for centralized logging and monitoring.

8. User Interface

- **Dashboard:** A web-based dashboard using frameworks like React or Angular to visualize predictions, model performance, and feature importance.
- **Access Control:** Implement authentication and authorization mechanisms to control access to the dashboard and APIs.