Adult Census Income Prediction

Detailed Project Report

Rushikesh Shinde

Data Science Intern at Ineuron.ai

Introduction

Predicting income levels based on demographic attributes is a fundamental task in socioeconomic research, with implications for policy-making, resource allocation, and understanding societal dynamics. The Adult Census Income Prediction dataset provides a rich source of anonymized information about individuals' demographics, such as age, education, occupation, and marital status, along with a binary income class label indicating whether an individual earns more than $50,000 annually.

This dataset has garnered significant attention in the machine learning community due to its relevance and complexity. Researchers have explored various predictive modeling approaches to accurately classify individuals into income categories based on their demographic characteristics. The challenge lies in effectively utilizing these demographic features to build models that generalize well to unseen data and provide meaningful insights into income disparities.

In this context, the Adult Census Income Prediction dataset serves as a benchmark for evaluating the performance of different machine learning algorithms, feature engineering techniques, and model evaluation metrics. By analyzing this dataset, researchers aim to uncover underlying patterns and factors that influence income distribution, ultimately contributing to a better understanding of socioeconomic dynamics.

This introduction sets the stage for exploring the Adult Census Income Prediction dataset and highlights its significance in advancing research on income prediction and socioeconomic inequality. Through comprehensive analysis and modeling efforts, researchers strive to develop robust predictive models that can inform policy decisions, drive targeted interventions, and promote economic empowerment for individuals and communities.

Objective

The primary objective of predicting income levels using the Adult Census Income Prediction dataset is to develop accurate and reliable models that can classify individuals into income categories based on their demographic attributes. This task serves several important purposes:

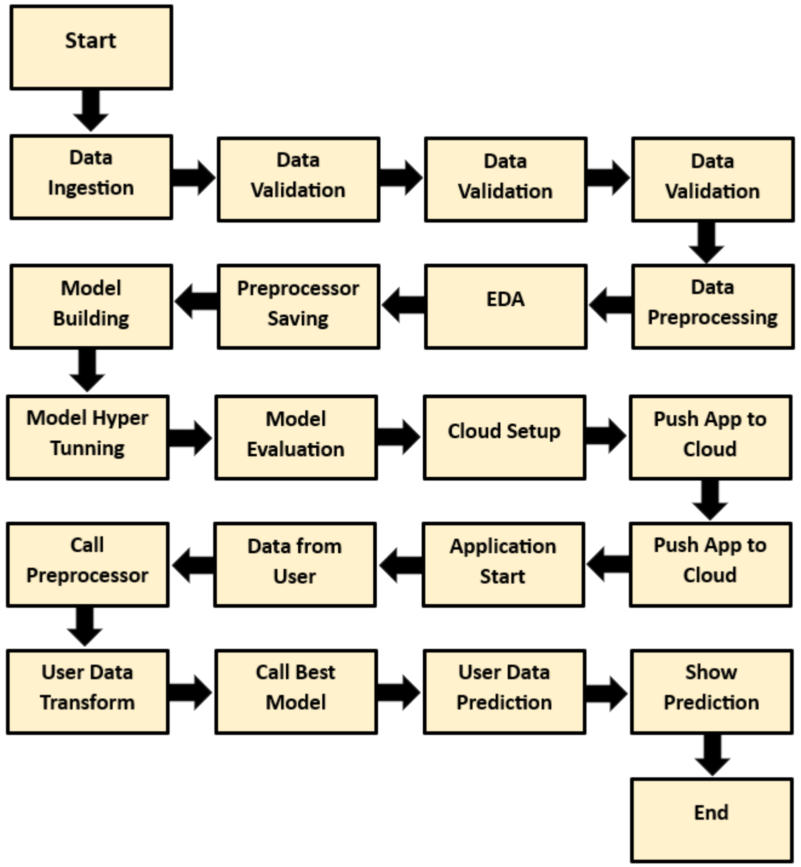
Socioeconomic Research: By understanding the relationship between demographic factors and income levels, researchers can gain insights into socioeconomic disparities, identify vulnerable populations, and explore the drivers of economic inequality.

Policy Making: Predictive models for income levels can inform the design and implementation of social welfare programs, taxation policies, and workforce development initiatives. Decision-makers can use these models to target resources effectively and address the needs of disadvantaged groups.

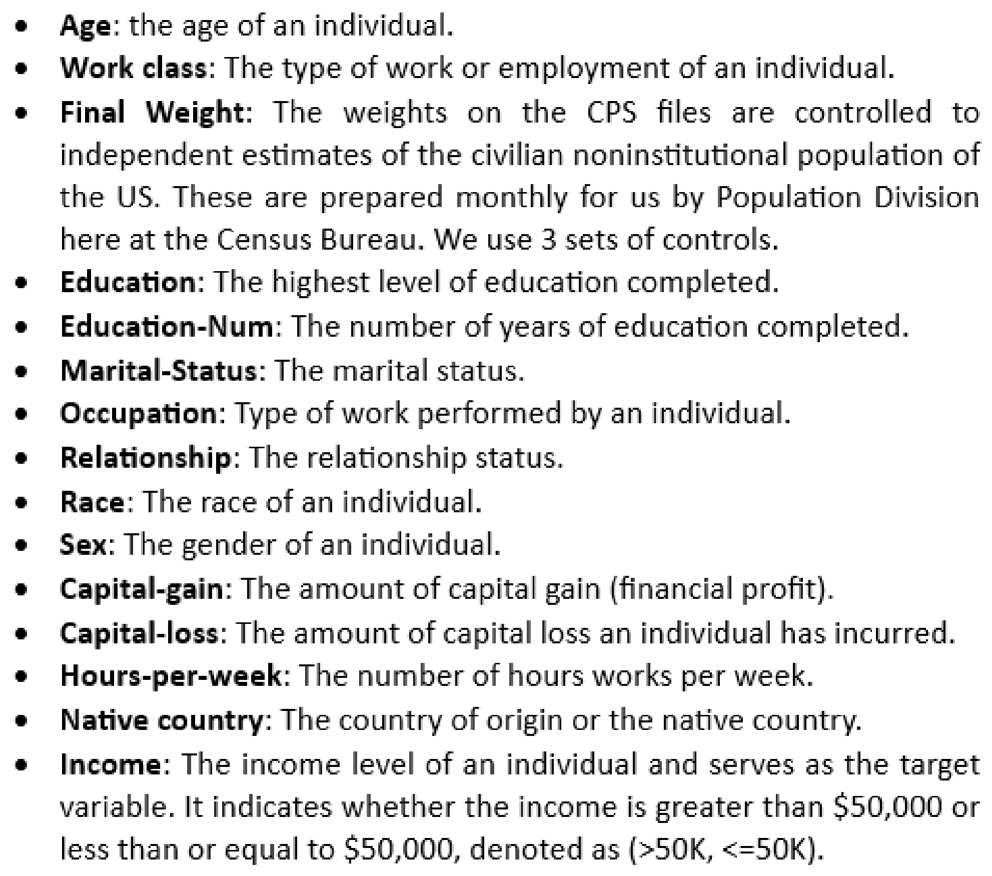
Resource Allocation: Predicting income levels enables organizations and governments to allocate resources efficiently. By identifying individuals or communities with lower incomes, resources such as education, healthcare, and housing can be directed to where they are most needed.

Business Applications: In the business sector, predicting income levels can aid companies in customer segmentation, marketing strategies, and product development. Understanding the income distribution of target markets helps businesses tailor their offerings to meet the needs and preferences of different consumer segments.

Architechture

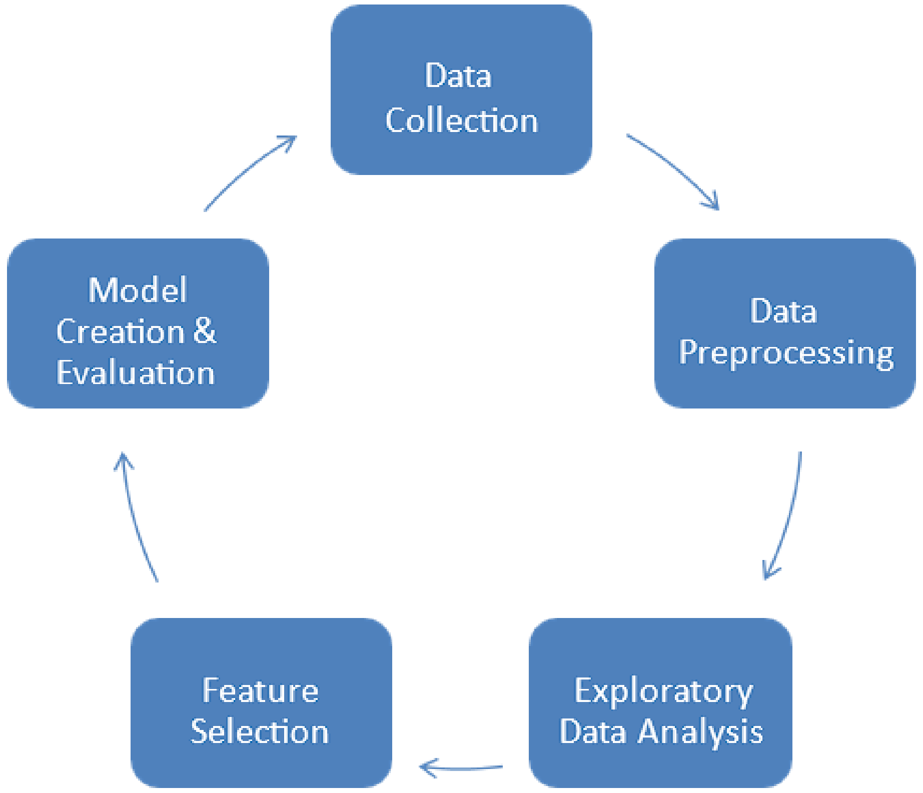


Dataset

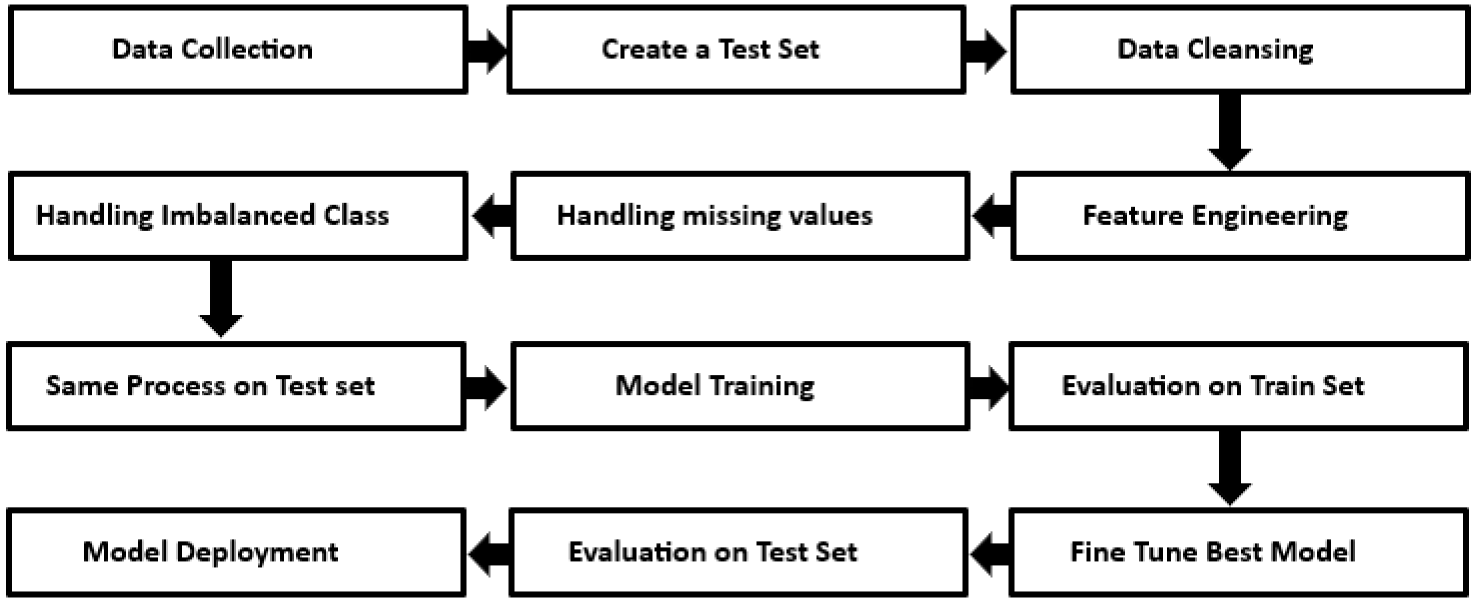


- **Age**: the age of an individual.
- **Work class**: The type of work or employment of an individual.
- **Final Weight**: The weights on the CPS files are controlled to independent estimates of the civilian noninstitutional population of the US. These are prepared monthly for us by Population Division here at the Census Bureau. We use 3 sets of controls.
- **Education**: The highest level of education completed.
- **Education-Num**: The number of years of education completed.
- **Marital-Status**: The marital status.
- **Occupation**: Type of work performed by an individual.
- **Relationship**: The relationship status.
- **Race**: The race of an individual.
- **Sex**: The gender of an individual.
- **Capital-gain**: The amount of capital gain (financial profit).
- **Capital-loss**: The amount of capital loss an individual has incurred.
- **Hours-per-week**: The number of hours works per week.
- **Native country**: The country of origin or the native country.
- **Income**: The income level of an individual and serves as the target variable. It indicates whether the income is greater than $50,000 or less than or equal to $50,000, denoted as (>50K, <=50K).

Data Analysis Steps



Model Training and Validation Workflow



Model Training and Validation Workflow

Data Collection

- For Data Set: https://www.kaggle.com/overload10/adult-census-dataset

Data Preprocessing

- Replacing missing values with nan values.
- Handing missing values with simple imputation.
- Categorical feature handing with one hot encoding.
- Feature selection using random forest classifier.
- Drop unnecessary columns with respect to importance by random forest.
- Handing Imbalanced data with SMOTE over-sampling.

Model Training and Validation Workflow

Model Creation and Evaluation

- Various classification algorithms like Logistic Regression, Decision Tree, Random Forest, AdaBoost, Gradient Boost, KNN and XGBoost trained and tested on both train and test set.
- Hyperparameter tunning on all the models in one go designed with python modular coding for efficient execution of model training and testing.
- Model performance evaluation with all the matrix for all the classes in one go again using python modular coding with use of classification report.
- Models can be tunned and evaluated in real time by changing the parameters in params.yaml file and execution model build file in one go the only requirement is to have a good CPU for fast processing and results.
- In model building and evaluation process for training the Decision tree and Random forest where preforming good with 100% accuracy meaning these models were over fitting as it was proven in the evaluation stage and Gradient Boost was performing the best in terms all the matrix for both the classes with 84% accuracy and 83% f1-score.

Gradient Boost Classifier

Introduction

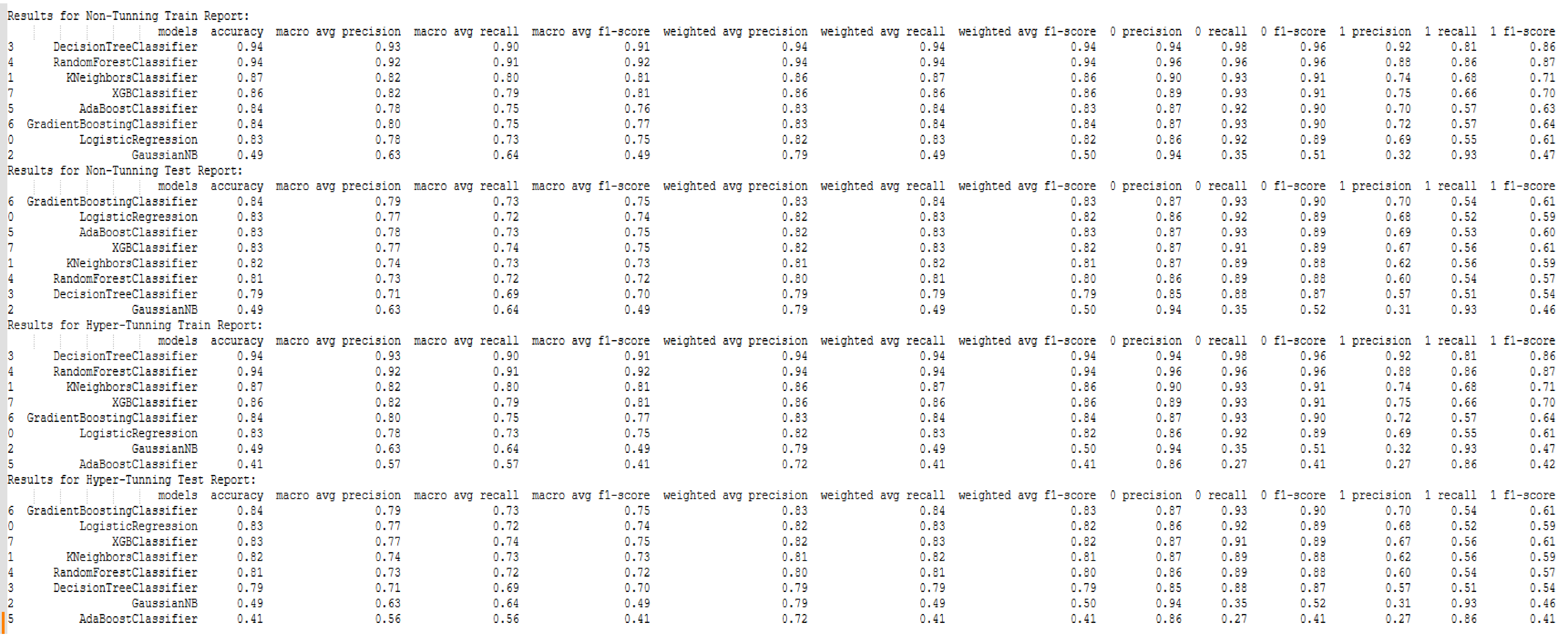
The Gradient Boost Classifier is a supervised machine learning algorithm which we can use for regression and classification problems. It is among the most popular machine learning algorithms comes under boosting ensemble technique.

Gradient Boost Classifier being ensemble algorithm tends to give more accurate result. This is because it works on the principle i.e. number of weak estimators when combined forms strong estimator. Even if one or few decision tree are prone to noise, overall results would tend to be correct.

Reason to use XGBoost Classifier model:

- The model learns from its mistakes and gives good results with respect to data provided.
- The calculated weights while model training allows it for prediction of new data with less error.

Training and Evaluation Results



```
Results for Non-Tunning Train Report:
                       models  accuracy  macro avg precision  macro avg recall  macro avg f1-score  weighted avg precision  weighted avg recall  weighted avg f1-score  0 precision  0 recall  0 f1-score  1 precision  1 recall  1 f1-score
3      DecisionTreeClassifier      0.94                 0.93              0.90                0.91                    0.94                 0.94                   0.94         0.94      0.98        0.96         0.92      0.81        0.86
4      RandomForestClassifier      0.94                 0.92              0.91                0.92                    0.94                 0.94                   0.94         0.96      0.96        0.96         0.88      0.86        0.87
1        KNeighborsClassifier      0.87                 0.82              0.80                0.81                    0.86                 0.87                   0.86         0.90      0.93        0.91         0.74      0.68        0.71
7               XGBClassifier      0.86                 0.82              0.79                0.81                    0.86                 0.86                   0.86         0.89      0.93        0.91         0.75      0.66        0.70
5          AdaBoostClassifier      0.84                 0.78              0.75                0.76                    0.83                 0.84                   0.83         0.87      0.92        0.90         0.70      0.57        0.63
6  GradientBoostingClassifier      0.84                 0.80              0.75                0.77                    0.83                 0.84                   0.84         0.87      0.93        0.90         0.72      0.57        0.64
0          LogisticRegression      0.83                 0.78              0.73                0.75                    0.82                 0.83                   0.82         0.86      0.92        0.89         0.69      0.55        0.61
2                  GaussianNB      0.49                 0.63              0.64                0.49                    0.79                 0.49                   0.50         0.94      0.35        0.51         0.32      0.93        0.47
Results for Non-Tunning Test Report:
                       models  accuracy  macro avg precision  macro avg recall  macro avg f1-score  weighted avg precision  weighted avg recall  weighted avg f1-score  0 precision  0 recall  0 f1-score  1 precision  1 recall  1 f1-score
6  GradientBoostingClassifier      0.84                 0.79              0.73                0.75                    0.83                 0.84                   0.83         0.87      0.93        0.90         0.70      0.54        0.61
0          LogisticRegression      0.83                 0.77              0.72                0.74                    0.82                 0.83                   0.82         0.86      0.92        0.89         0.68      0.52        0.59
5          AdaBoostClassifier      0.83                 0.78              0.73                0.75                    0.82                 0.83                   0.83         0.87      0.93        0.89         0.69      0.53        0.60
7               XGBClassifier      0.83                 0.77              0.74                0.75                    0.82                 0.83                   0.82         0.87      0.91        0.89         0.67      0.56        0.61
1        KNeighborsClassifier      0.82                 0.74              0.73                0.73                    0.81                 0.82                   0.81         0.87      0.89        0.88         0.62      0.56        0.59
4      RandomForestClassifier      0.81                 0.73              0.72                0.72                    0.80                 0.81                   0.80         0.86      0.89        0.88         0.60      0.54        0.57
3      DecisionTreeClassifier      0.79                 0.71              0.69                0.70                    0.79                 0.79                   0.79         0.85      0.88        0.87         0.57      0.51        0.54
2                  GaussianNB      0.49                 0.63              0.64                0.49                    0.79                 0.49                   0.50         0.94      0.35        0.52         0.31      0.93        0.46
Results for Hyper-Tunning Train Report:
                       models  accuracy  macro avg precision  macro avg recall  macro avg f1-score  weighted avg precision  weighted avg recall  weighted avg f1-score  0 precision  0 recall  0 f1-score  1 precision  1 recall  1 f1-score
3      DecisionTreeClassifier      0.94                 0.93              0.90                0.91                    0.94                 0.94                   0.94         0.94      0.98        0.96         0.92      0.81        0.86
4      RandomForestClassifier      0.94                 0.92              0.91                0.92                    0.94                 0.94                   0.94         0.96      0.96        0.96         0.88      0.86        0.87
1        KNeighborsClassifier      0.87                 0.82              0.80                0.81                    0.86                 0.87                   0.86         0.90      0.93        0.91         0.74      0.68        0.71
7               XGBClassifier      0.86                 0.82              0.79                0.81                    0.86                 0.86                   0.86         0.89      0.93        0.91         0.75      0.66        0.70
6  GradientBoostingClassifier      0.84                 0.80              0.75                0.77                    0.83                 0.84                   0.84         0.87      0.93        0.90         0.72      0.57        0.64
0          LogisticRegression      0.83                 0.78              0.73                0.75                    0.82                 0.83                   0.82         0.86      0.92        0.89         0.69      0.55        0.61
2                  GaussianNB      0.49                 0.63              0.64                0.49                    0.79                 0.49                   0.50         0.94      0.35        0.51         0.32      0.93        0.47
5          AdaBoostClassifier      0.41                 0.57              0.57                0.41                    0.72                 0.41                   0.41         0.86      0.27        0.41         0.27      0.86        0.42
Results for Hyper-Tunning Test Report:
                       models  accuracy  macro avg precision  macro avg recall  macro avg f1-score  weighted avg precision  weighted avg recall  weighted avg f1-score  0 precision  0 recall  0 f1-score  1 precision  1 recall  1 f1-score
6  GradientBoostingClassifier      0.84                 0.79              0.73                0.75                    0.83                 0.84                   0.83         0.87      0.93        0.90         0.70      0.54        0.61
0          LogisticRegression      0.83                 0.77              0.72                0.74                    0.82                 0.83                   0.82         0.86      0.92        0.89         0.68      0.52        0.59
7               XGBClassifier      0.83                 0.77              0.74                0.75                    0.82                 0.83                   0.82         0.87      0.91        0.89         0.67      0.56        0.61
1        KNeighborsClassifier      0.82                 0.74              0.73                0.73                    0.81                 0.82                   0.81         0.87      0.89        0.88         0.62      0.56        0.59
4      RandomForestClassifier      0.81                 0.73              0.72                0.72                    0.80                 0.81                   0.80         0.86      0.89        0.88         0.60      0.54        0.57
3      DecisionTreeClassifier      0.79                 0.71              0.69                0.70                    0.79                 0.79                   0.79         0.85      0.88        0.87         0.57      0.51        0.54
2                  GaussianNB      0.49                 0.63              0.64                0.49                    0.79                 0.49                   0.50         0.94      0.35        0.52         0.31      0.93        0.46
5          AdaBoostClassifier      0.41                 0.56              0.56                0.41                    0.72                 0.41                   0.41         0.86      0.27        0.41         0.27      0.86        0.41
```

Deployment

Model Deployment

We will be deploying the model to AWS.

Frequently Asked Questions

1) **What is the source of data?**

   The data for training is obtained from famous machine learning repository.

   Kaggle Machine Learning Repo: https://www.kaggle.com/overload10/adult-census-dataset

2) **What was the type of data?**

   The data consists of both numerical and categorical features with continuous and desecrate values.

3) **What's the complete flow you followed in this Project?**

   Refer file 4, 5, 6 and 7 for more better understanding.

4) **How are logs managed?**

   We are using different logs as per the steps that we follow in training and prediction like model training log and prediction log etc. And then sub log are inside those folder.

Frequently Asked Questions

5) **What techniques were used for data-preprocessing?**
    - Removed unwanted features
    - Visualizing relation of independent variables with each other and output variable.
    - Handing missing values with simple imputation.
    - Converting categorical columns to numerical values.

6) **How training was done, or models where used?**
    - Training and evaluation was done with various models with python modular coding by creating custom function in combination with scikit-learn for train and test execution in one go with hyperparameter tunning.
    - Finally best model is selected based up the custom evaluation matrix.

7) **How was prediction done?**

    Prediction are done with the help of website build with HTML and CSS, with flask as backend for using the data and getting the prediction of user in real time eliminating the long process of human verification from experienced people to determine the income of an individual.