**Professor Viswanatha Rao (Vishwa)**

**Introduction to Machine Learning**

**Midterm-ITAI-1371-Group-4**

**Due October 28 / 2025**

**Prediction whether a person will be making more than $50k**

**Library used:**

matplotlib.pyplot, numpy , pandas,

sklearn.preprocessing import StandardScaler,

sklearn.preprocessing import MinMaxScaler

sklearn.preprocessing import OneHotEncoder, LabelEncoder

from sklearn.impute import SimpleImputer

**1.Data Loading & Exploration :** This is like **opening up the data file and looking through it** for the first time. check how many rows and columns, what kind of information is in each column (numbers, words, etc.)

**Code:** df / df.shape / df.info() / df.isnull() / df.isnull().sum()

**2. Removed Unnecessary and Constant Columns:**

Get rid of unnecessary columns to keep things clean. Separate Target variable from the dataset. Unnecessary columns can include:

* Identifier-like columns (e.g., 'id', 'fnlwgt' if not useful)
* Duplicate or constant-value columns
* Columns with too many unique values relative to dataset size
* Irrelevant for modeling (like names or timestamps)

Detect and remove such columns automatically.

* Check constant columns (only one unique value)
* Check quasi-identifiers (e.g., 'fnlwgt' in Adult dataset)
* 'fnlwgt' is a census weight — not useful for prediction.
* Drop the unnecessary columns

Separate Target Variable (predict income )

**3. Feature Engineering (Outliers, Binning, Domain Features)**

Manipulating the data to make it more useful for the model.

* **Outliers:** look for extreme data points and decide whether to fix them or remove them because they can confuse the model.
* **Binning:** This is like grouping numbers into ranges. For example, turning a person's exact age (25, 32, 48) into age groups (Young, Middle-aged, Senior).
* **Domain Features:** Using real-world knowledge (the "domain") to create new, helpful columns. E.g., Height and Weight, you can create a new feature called BMI.

**4. Filling NaN and Null Values (Handled Missing Values)**

some entries are often blank. These "Not a Number" (NaN) or **missing values** will break the model, so you have to **c**arefully fill them in

Strategy: Categorical: Fill with mode (most frequent) and Numerical: Fill with median

**5. One-Hot Encoding (Encoded Categorical Features)**

OneHot Encoder converting categorical txt to numbers. Computers like numbers, but your data might have words , I understood that **One-Hot Encoding** is how we **turn these categories into numbers** the model can understand, creating a clean DataFrame. Generating meaningful column names and storing the result in a new, manageable DataFrame for easy merging or concatenation with other features.

A screenshot of a computer code

AI-generated content may be incorrect.

I used this method for two things:

* **fit:** It learns the unique categories present in each column of X[categorical\_cols].
* **transform:** It applies the one-hot encoding logic to those columns, resulting in a NumPy array of 0s and 1s (because sparse\_output=False).
* **encoder.get\_feature\_names\_out(categorical\_cols):** After fitting, this method generates the new column names for the one-hot encoded data. The names follow a pattern like original\_col\_category\_value

**6. Scaling and Standardizing: Changing the range the data - Scaled and normalized data. Scaling Normalization: Adjusting the shape of the data's distribution**

This step is all about making sure all the numbers are **playing fair**. If one column is *Age* (0 to 100) and another is *Yearly Income* (0 to 1,000,000), the income column will totally dominate the model.

* **Scaling/Normalization:** **Squishes all the numbers into a small, consistent range** (like 0 to 1) so no single feature overpowers the others.

# Normalization: Adjust shape (MinMaxScaler)

# MinMax Normalized version

* **Standardization:** **Adjusts the data's shape** so that it has an average of 0 and a consistent spread, making it easier for some models to learn.

**7. Correlation-Based Feature Reduction**

**Correlation** measures how closely two variables are related. If *Years of Education* and *Degree Level* are almost identical in what they tell you, you only need to keep one. You use this analysis and a **visualization** (like a heat map) to **cut out highly related, redundant features** to speed up the model and keep it simple.

**Model Training & Evaluation**

**8. Train-Test Split**

Before training the model, you **split your data into two groups**:

1. **Train Set:** The large part the model **studies and learns** from.
2. **Test Set:** The smaller part the model has **never seen**, for later to see how well the model truly performs on new data.

**09. Random Forest Model Training**

The **Random Forest** is the predictor tool. It's an algorithm that builds **many individual "decision trees"** (like flowcharts) and has them all vote on the final answer. We may **Train Set** to "feed" this forest until it learns the patterns needed to make accurate predictions.

**10. Performance Evaluation and ROC**

Project: Predict whether a person will be making more than $50k

A screenshot of a computer program

AI-generated content may be incorrect.

The code provides a solution to predict income bracket for making more than 50k using the adult.csv dataset,

covering all necessary steps from data cleaning and preprocessing to model training and evaluation using Logistic Regression.

Accuracy: The overall percentage of predictions the model got

Fi-Score: providing a balanced measure of performance.

AUC means evaluation. The model has a high ability to distinguish between the two income classes.

**After training, I use test set to see how good the model is.**

* **Evaluation:** You check the model's accuracy (how often it's right) and other scores.
* **ROC (Receiver Operating Characteristic):** This is a special way to measure the model's ability to correctly separate the two groups (making over $50k vs. making less). A perfect score is 1.0.

**Analysis & Final Goal**

**11. Income Distribution – Target Variable Analysis**

Before starting, it is a good idea to **look closely at the Target Variable** (the income groups). Check what percentage of people make *over* $50k versus *under* $50k. This helps to understand how balanced the prediction problem is and gives a **baseline** to beat.

A screenshot of a graph

AI-generated content may be incorrect.

**12. Prediction: Making More Than $50k income Distribution**

**Target Variable Analysis on Test Data**

* This is the **final goal**: Using the Random Forest model built, so I can now input a new person's details (age, education, job, etc.) and have the model **predict** whether that person will likely be **making more than $50,000** or not.

A screenshot of a computer

AI-generated content may be incorrect.

Works Cited

Google page: research, writing and spelling correction

Google Collab AI