Prepocessing

In your **pipeline**, missing values (? in the dataset) were treated as follows:

**Numeric features (e.g., Age, Hours-per-week, CapitalGains/Losses, WeightingFactor, SchoolingYears)**

* Imputation strategy: **Median**

("imputer", SimpleImputer(strategy="median"))

* Why median?
  + Robust against outliers (better than mean if data is skewed).
  + Ensures you don’t lose rows just because of missing numbers.

**Categorical features (e.g., EmploymentType, MaritalStatus, Occupation, CountryOfBirth)**

* Imputation strategy: **Most frequent (mode)**

("imputer", SimpleImputer(strategy="most\_frequent", fill\_value="Unknown"))

* Why?
  + Keeps categories consistent.
  + If a feature is missing, assigning it to the most common group or to "Unknown" prevents dropping data.

**After imputation:**

* **Categoricals** → encoded with OneHotEncoder(handle\_unknown="ignore")
* **Numerics** → scaled with StandardScaler (except when using Random Forest, scaling doesn’t matter much, but the pipeline keeps it consistent).

So:

* Missing numeric values get filled with a **representative number (median)**.
* Missing categorical values get filled with the **most frequent category (or “Unknown”)**.

Model choosing 1st:

**🔹 Logistic Regression**

* **What it does:**  
  A **linear classifier** that models the probability of the income label (<=50K or >50K) using the logistic (sigmoid) function.
* **Why it fits your dataset:**
  + Your features are mostly **categorical (occupation, marital status, country, etc.)**, which can be easily one-hot encoded. Logistic regression works well with **high-dimensional sparse inputs**.
  + It produces **probability estimates** (e.g., "this person has 78% chance of earning >50K"), which is useful when interpreting income predictions.
  + Fast to train, interpretable, and scales well to **30,000 rows**.

**🔹 Support Vector Machines (SVM)**

* **What it does:**  
  Finds the hyperplane that best separates the two income groups, possibly in a higher-dimensional space via **kernels**.
* **Pros:**
  + Very strong at finding complex non-linear decision boundaries.
  + Robust to high-dimensional data.
* **Cons for your case:**
  + Training time grows **quadratically** with dataset size — not great for 30k rows.
  + Memory intensive (needs kernel matrix).
  + Harder to interpret compared to logistic regression.
  + Probabilities are not natural outputs (need Platt scaling or calibration).

**🔹 Why Logistic Regression is usually chosen here**

* In applied projects like income prediction:
  + **Interpretability** matters: we want to explain which features (e.g., education, hours worked) contribute to income differences. Logistic regression provides **feature weights**.
  + **Scalability:** Training logistic regression is much faster than training SVMs on 30k+ samples.
  + **Baseline model:** Logistic regression is a strong baseline and often competitive with more complex models.

In fact, your **Bayesian Learning slides** even connect logistic regression with **probabilistic classification**, reinforcing why it is a standard choice for binary labels like income.

**Why Random Forest could be chosen**

* **Non-linear relationships:** Unlike Logistic Regression (linear), Random Forest can capture complex interactions (e.g., “married + certain occupation + age range” patterns).
* **Robust to irrelevant features:** Trees ignore unhelpful features during splitting, so less need for strict feature selection.
* **Handles categorical features well after one-hot encoding**.
* **Works well with mixed data types:** numerical + categorical.
* **Less sensitive to scaling:** Unlike Logistic Regression or SVM, you don’t need to normalize inputs for trees.
* **Interpretability via feature importance:** You can report which features matter most (education, working hours, marital status).

**⚠️ Downsides compared to Logistic Regression**

* Slower to train/predict (though with 5k samples, not a big issue).
* Harder to interpret exact coefficients compared to Logistic Regression.
* Provides probabilities, but they are less well-calibrated.

👉 In your case:

* **Logistic Regression** was chosen because it had a *slightly better F1* on your test split.
* But **Random Forest is still a valid choice** — you could justify it by saying it captures more complex patterns, and show its feature importance.
* In an exam/project report, you might even say:  
  *“We compared Logistic Regression and Random Forest. Logistic Regression had slightly higher F1, but Random Forest offers richer modeling of non-linearities. Both are valid; we continued with Logistic Regression for simplicity and interpretability.”*

Results:

**1. Data check**

Total rows : 30000

Labeled rows : 5000 (expected ~5000)

Unknown rows : 25000 (expected ~25000)

Confirms the dataset split worked:

* **5,000 labeled rows** → used for training/testing
* **25,000 unlabeled rows (Income=?)** → will be predicted

**2. Model evaluation (on a 1,000-sample test split from the 5k labeled data)**

You tested **two models**: Logistic Regression and Random Forest.

**Logistic Regression**

accuracy: 0.857

precision: 0.776

recall: 0.582

f1: 0.665

roc\_auc: 0.905

* **Accuracy (85.7%)**: Overall, ~86% of predictions are correct.
* **Precision (0.776)**: When the model predicts >50K, it’s correct ~77.6% of the time.
* **Recall (0.582)**: The model finds ~58% of the actual >50K people (it misses some high-income cases).
* **F1 (0.665)**: Balance of precision & recall → ~66%.
* **ROC-AUC (0.905)**: Very good! It shows the model separates the two classes well (much better than random guessing at 0.5).

Interpretation: **LogReg is conservative** — it’s better at spotting <=50K but misses some >50K.

**Random Forest**

accuracy: 0.843

precision: 0.712

recall: 0.598

f1: 0.650

roc\_auc: 0.892

* Slightly lower **accuracy (84.3%)**.
* **Precision (71%)** is a bit lower than LogReg.
* **Recall (59.8%)** is slightly higher (finds a few more high-income people).
* **F1 (0.650)** is slightly worse than LogReg.
* **ROC-AUC (0.892)** is still strong, but lower than LogReg.

Interpretation: **RandomForest is a bit more balanced**, but overall performed slightly worse in F1.

**3. Best Model Selection**

Best by F1: LogisticRegression (F1=0.665)

* The code chose **Logistic Regression** as the final model because it had the **highest F1-score**.
* F1 was chosen over accuracy because your dataset is **imbalanced** (fewer people >50K). F1 is better for evaluating performance on the minority class.

**4. Predictions on unlabeled data**

Saved predictions → outputs/predictions\_25000.csv

* Your model predicted incomes for **all 25,000 unlabeled rows**.
* The file contains PredictedIncome (<=50K / >50K) and probability (if available).

**5. Processed matrices for correlation/regression**

Processed matrices saved in outputs

* The script exported:
  + processed\_known\_features.csv → numeric-only matrix for the 5,000 labeled rows
  + processed\_unknown\_features.csv → numeric-only matrix for the 25,000 unlabeled rows
* These are ready for **correlation/regression analysis** (part of your assignment).

**6. Model saved**

Saved trained model → outputs/best\_income\_model.joblib

* Your trained Logistic Regression pipeline (preprocessing + model) is saved with joblib.
* You can reload it later without retraining.

**summary:**

* Logistic Regression was chosen as the best model (F1 = 0.665).
* Predictions for 25k unknown incomes were saved.
* You have clean feature matrices for further analysis.
* Model is stored for reuse.

**Confusion matrix**

shows where errors happen (false positives vs. false negatives). That’s crucial with class imbalance (your >50K class is the minority).

**ROC curve**

shows performance across thresholds, not just at 0.5. Together with **ROC-AUC** it helps argue whether you should tweak the decision threshold for better recall/precision trade-offs.