

Machine Learning & Neural Networks

Access more content here: [*GitHub – Machine Learning and Neural Networks*](#)

Random Forest Tutorial

By
Awais Mumtaz ul Haq
ID: 23096809

Random Forest

1. Random Forest:

1.1 Introduction:

Random forests are supervised machine learning algorithms. It features two important variations: one for classification and the other for regression. It is one of the most versatile and user-friendly algorithms. It generates decision trees based on the data samples provided, extracts predictions from each tree, and votes on the best answer. It is also a fairly effective predictor of feature importance.

- The **Classification Task** in Random Forest is completed by aggregating the votes of many decision trees.
- The **regression task** in Random Forest is completed by averaging the predictions of each individual tree.

The Random Forest algorithm mixes numerous decision trees, resulting in a forest of trees, as the name implies. In the random forest classifier, the number of trees in the forest correlates with higher accuracy.

Decision trees present you with a difficult choice. A deep tree with many leaves will overfit since each forecast is based on past data from only a few residences on each leaf. However, a shallow tree with few leaves will perform poorly since it cannot capture as many differences in the raw data.

Even today's most sophisticated modelling approaches face the tension between underfitting and overfitting. However, many models incorporate creative ideas that can lead to improved performance. We'll use the random forest as an example.

1.2 Random Forest Algorithm Intuition

Stage 1: Create the Random Forest

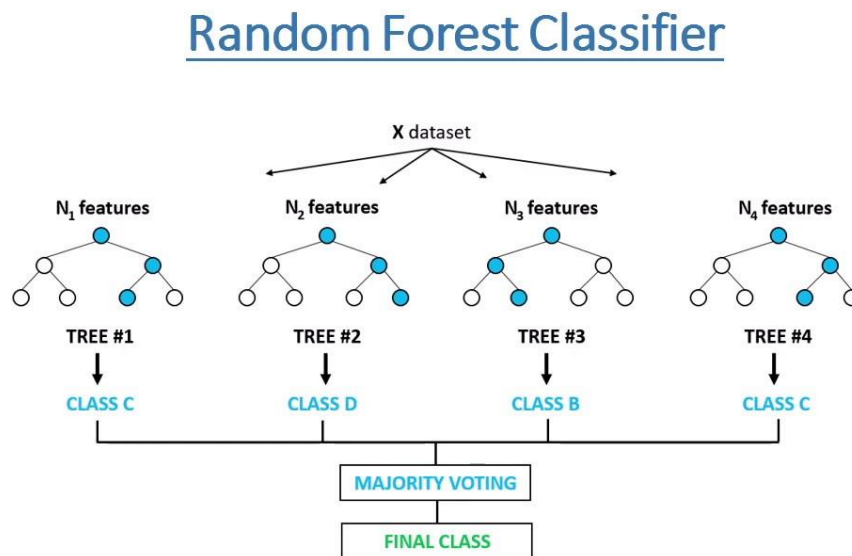
1. Choose k features randomly from a total of m features ($k < m$).
2. Choose the optimal split point from the specified k features to form a decision node (d).
3. Divide the node into daughter nodes depending on the optimal split criterion.
4. Repeat steps 1–3 recursively until a certain number of nodes (l) are attained.
5. Repeat steps 1–4 n times to generate n decision trees, resulting in the random forest.

Stage Two: Making Predictions

1. Process each test sample's features through all n decision trees in the forest.
2. Each tree predicts a certain result (class label or regression value).
3. Collect predictions from each tree.
4. To classify:
 - Count votes for each class.
 - The class receiving the most votes is chosen as the final prediction.
5. In regression, calculate the average (or weighted average) of all tree predictions to determine the final result.

This technique improves accuracy and reduces overfitting when compared to individual decision trees.

1.3. Random Forest Algorithm Intuition (Diagram):



1.4 Understanding Random Forest: Internal Mechanisms and Variants.

1.4.1. Random Forest's Internal Workings

- The method aggregates predictions from numerous decision trees to improve accuracy and reduce overfitting.
- The Bagging (Bootstrap Aggregating) approach creates various subsets of data by sampling and replacing the original dataset.
- Decision trees are trained independently on distinct bootstrap samples to ensure model variety.
- During training, a random subset of features is split at each node of a tree to avoid relying on dominant features and reduce correlation among trees.
- After training all trees, predictions are aggregated using majority voting (classification) or averaging (regression) to achieve a stable final result.

1.4.2. Key Internal Mechanisms of Random Forest

- Random Forest randomly selects a subset of features at each decision node, resulting in various tree architectures.
- Random Forest improves generalisation by merging weak learners, reducing overfitting compared to deep decision trees.
- Handling missing values: It can estimate missing values via proximity-based imputation, which improves its robustness in real-world scenarios.
- Out-of-Bag (OOB) Error Estimation: Since each tree is trained on a different subset, unused samples (OOB samples) aid in assessing model performance without a separate validation set.

1.4.3. Variants of Random Forest:

- **Extra Trees (Extremely Randomised Trees):** Increases training speed and reduces variance by randomly selecting split spots rather than the best split.
- **Weighted Random Forest** optimises model performance by assigning varying weights to trees based on predictive power.
- **The Random Subspace Method** trains trees using randomly selected subsets of features instead of bootstrapping data, making it appropriate for high-dimensional datasets.
- **Oblique Random Forest** improves decision-making in complex datasets by separating features based on linear combinations rather than single thresholds.

1.5 Advantages and Disadvantages of Random Forest Algorithm

1.5.1 Advantages:

- Supports both classification and regression, making it suitable for various machine learning workloads.
- High Accuracy and Robustness: Multiple decision trees reduce variation and improve forecasts.
- Resistant to overfitting: Predictions are averaged across trees to eliminate individual biases.
- Effectively addresses missing values.
- Replace missing continuous data with the median.
- Apply proximity-weighted average to estimate missing values.
- Improves model efficiency by identifying key features in the dataset.

1.5.2 Disadvantages:

- High computational complexity.
- Slow training and prediction due to the high number of decision trees.
- Real-time forecasts are wasteful due to the need for each tree to develop a prediction and then vote on it.
- Random forests are not as transparent as single decision trees, making interpretation challenging.
- Understanding the process of making individual predictions is challenging.
- Feature Selection with Random Forests

1.6 How Feature Importance is Measured

Random Forests can help with feature selection by ranking the importance of variables in a classification or regression problem. This aids in picking the most relevant features while discarding less important ones, hence increasing model efficiency and performance.

1. Train the Random Forest Model:

- Apply the Random Forest algorithm on the dataset.
- Note the Out-of-Bag (OOB) error for each data point.
- Calculate the average OOB error for all trees in the forest.

2. Calculate the Importance of Features:

- Randomly permute the j-th feature's values throughout the training data.
- Recalculate the OOB error using the changed dataset.
- Calculate the difference between the original and new OOB errors.

- Average the difference between all trees in the forest.
- Normalise the importance score using the standard deviation of the differences.

3. Feature Ranking and Selection:

- Features with higher significance scores are deemed more significant.
- Simplify the model by removing lower-scoring features.

This strategy ensures that only the most important features are employed, resulting in a more interpretable and efficient model with strong predictive performance.

1.7 Difference Between Random Forests and Decision Trees

Random Forests and Decision Trees are two frequently used machine learning algorithms, although they differ in several important ways. This is a structured comparison:

1. Structure.

- A decision tree is a hierarchical model that makes decisions based on feature splits.
- Random Forest is an ensemble of decision trees that combine their outputs for more accurate predictions.

2. Computational Speed

- The Decision Tree algorithm is faster for training and prediction due to its single tree structure.
- Random Forest is slower as it requires training numerous trees and aggregating their results.

3. Overfitting.

- Deep decision trees can overfit training data, resulting in poor generalisation to fresh data.
- Random Forest: Reduces overfitting by averaging many decision trees, resulting in better model stability.

4. Interpretability.

- Decision Trees are easily interpretable and may be transformed into decision rules.
- Random Forest: Multiple trees operate together, making it more complex and difficult to grasp.

5. Performance for Large Datasets

- Decision Trees perform well on smaller datasets but may struggle with larger ones due to overfitting.
- Random Forest optimises massive datasets by reducing volatility and enhancing accuracy.

6. Handling Missing Data.

- Decision Tree: While capable of handling missing values, it lacks robustness.
- Random Forest excels at handling missing values through approaches like proximity-weighted imputation.

2.1 Problem Statement:

The purpose of this project is to estimate whether a person makes more than \$50,000 per year using Random Forest Classification with Python and Scikit-Learn.

In this project, I created two Random Forest Classifier models to predict if a person earns more than \$50,000 per year: one with 10 decision trees (Default) and the other with 100 decision trees. The estimated accuracy improves with the number of decision trees in the model. I illustrated the feature selection process using the Random Forest model to identify only the most essential elements, rebuild the model with these features, and assess the influence on accuracy. I used the Income categorisation data set for this project.

3. Implementation in python:

This project's dataset, labelled "Income Evaluation," came from Kaggle. This dataset includes demographic and employment information, making it appropriate for predictive modelling applications like income classification. Dataset contains 32561 instances (Rows) and 15 attributes (Columns)

3.1 (EDA) Exploratory data analysis:


3.1.1 Summary of Dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   age                   32561 non-null  int64
1   workclass              32561 non-null  object
2   fnlwgt                 32561 non-null  int64
3   education              32561 non-null  object
4   education_num          32561 non-null  int64
5   marital_status         32561 non-null  object
6   occupation             32561 non-null  object
7   relationship           32561 non-null  object
8   race                   32561 non-null  object
9   sex                    32561 non-null  object
10  capital_gain            32561 non-null  int64
11  capital_loss            32561 non-null  int64
12  hours_per_week          32561 non-null  int64
13  native_country          32561 non-null  object
14  income                  32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

- The dataset includes 9 categorical categories and 6 numerical features.
- This analysis focusses on `income`, categorising individuals based on their incomes.
- So far, no missing values have been found; nevertheless, a more thorough investigation will be done to ensure data completeness.

3.1.2 Statistical Analysis:

Statistical Analysis is done on Numerical Features by Executing `.describe()` Method. Statistical Analysis on all Features can be done by Executing `df.describe(include='all')` Method.



	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

The dataset consists of 32,561 records and six numerical characteristics. The average age is 38.58 years, and most people work 40 to 45 hours per week. Capital gains and losses show considerable outliers, with capital gains peaking at 99,999. Education ranges from 1 to 16 years, with a median of 10 years, indicating regular academic growth.

3.1.3 Preview Categorical Features:

- The dataset contains nine categorical variables (features).
- The categorical variables (Features) include 'workclass', 'education', 'marital_status', 'occupation', 'relationship', 'race', 'sex', 'native_country', and 'income'.
- The goal variable is 'income' (feature).

3.1.4 Encoding Categorical Values:

As we know that Machine Learning Model only Works with Numerical Data. So we are required to Convert Categorical Values to Numerical Values for Robust Training.

```
# encode categorical variables with one-hot encoding

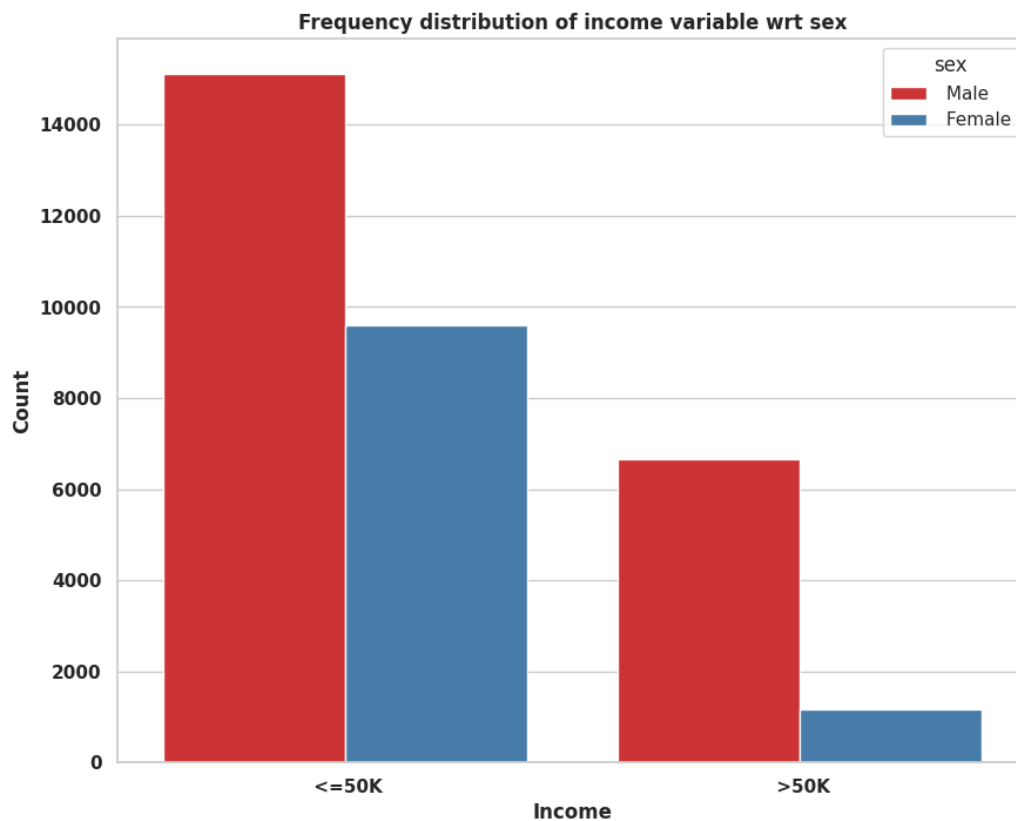
encoder = ce.OneHotEncoder(cols=['workclass', 'education', 'marital_status', 'occupation', 'relationship',
                                'race', 'sex', 'native_country'])

X_train = encoder.fit_transform(X_train)

X_test = encoder.transform(X_test)
```

4. Visualizations of Dataset:

4.1 Frequency Distribution of Income by Sex (Gender)



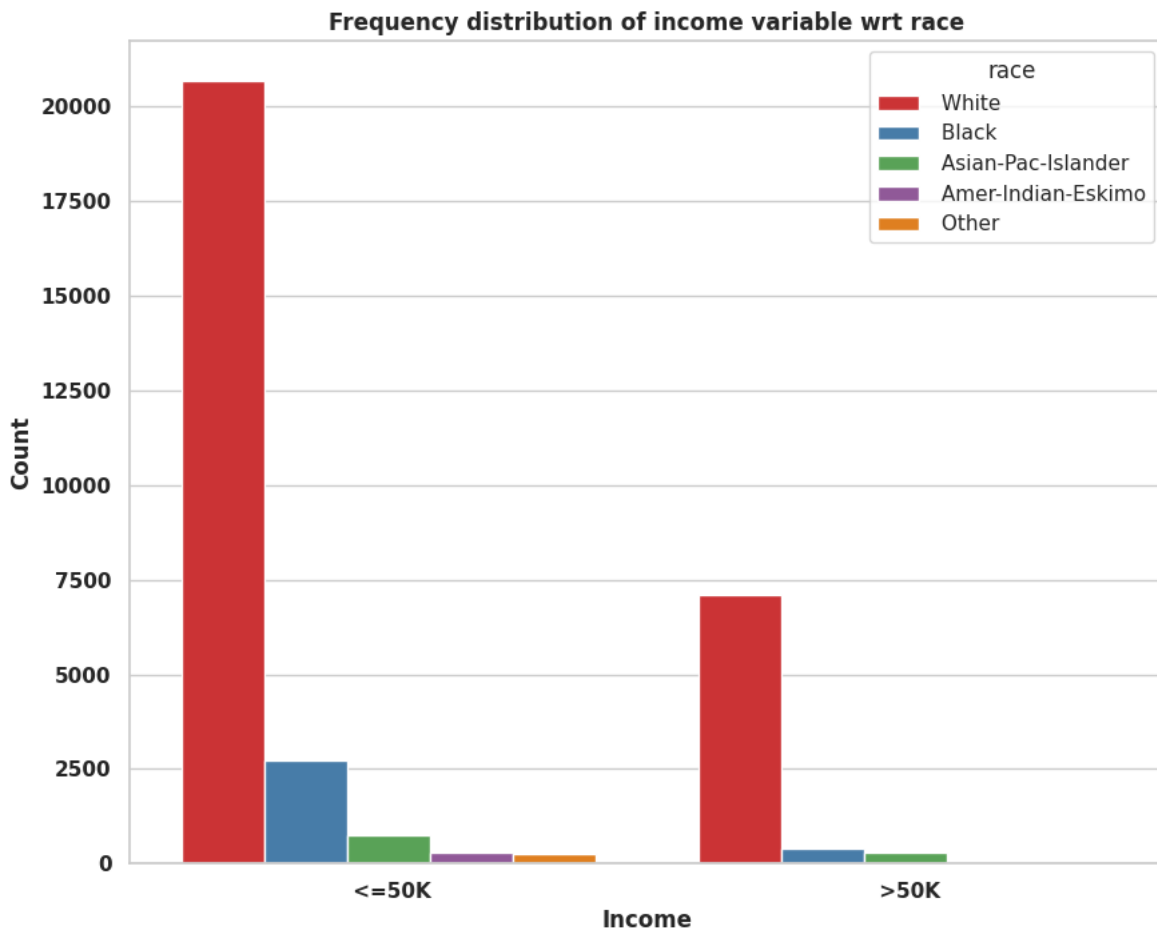
The bar chart represents the frequency distribution of income levels (<=50K and >50K) with respect to gender (Male and Female).

Observations:

- A significantly higher number of males and females fall into the <=50K income category, with males having the highest count.
- The >50K income category has fewer individuals overall, with males outnumbering females by a large margin.
- The income disparity is evident, as the number of males earning >50K is substantially higher than that of females.
- The gap between male and female counts is larger in the higher-income group, indicating a gender-based difference in earnings.

This visualization highlights income distribution disparities based on gender in the dataset.

4.2 Frequency Distribution of Income by Race



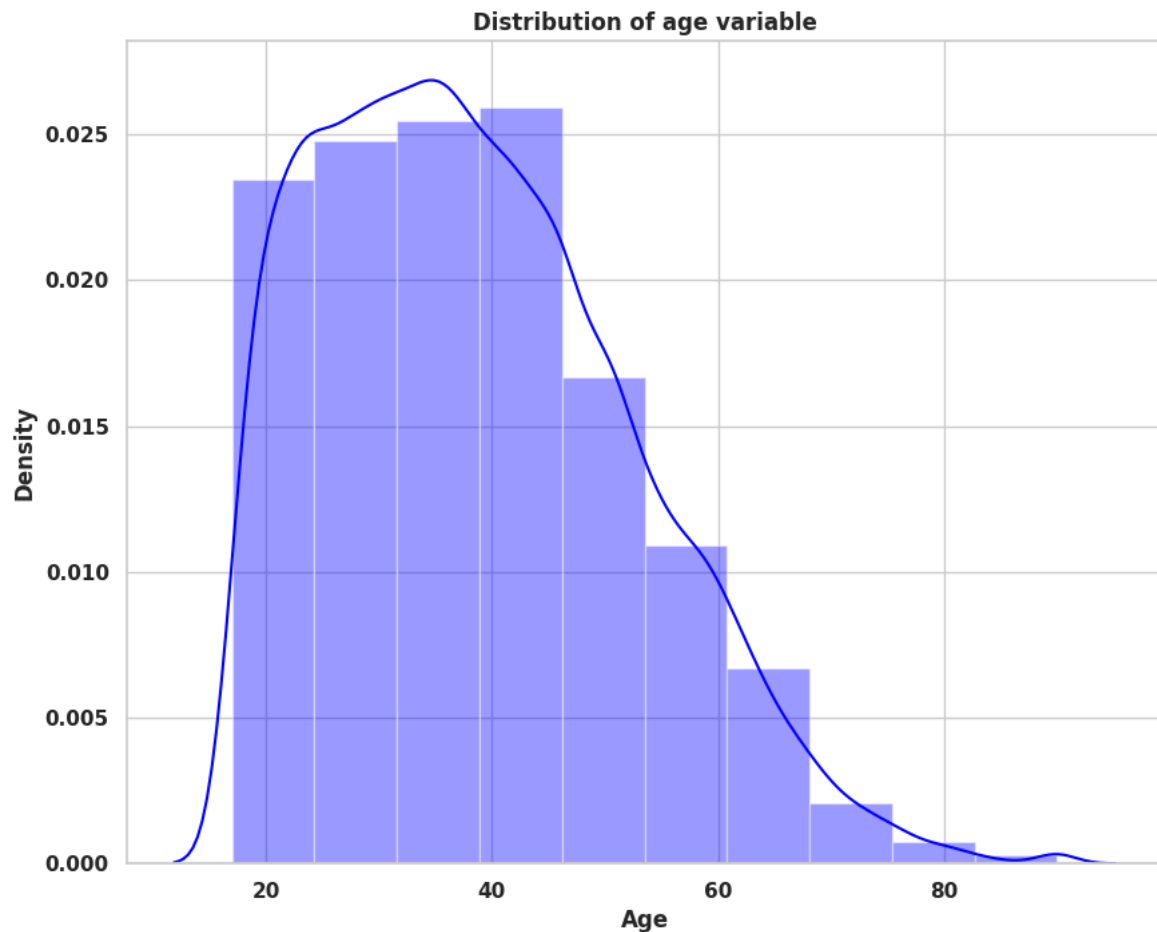
The bar chart illustrates the frequency distribution of income levels (<=50K and >50K) across different racial groups.

Observations:

- The majority of individuals in the dataset belong to the White racial group, making up the largest portion in both income categories.
- The <=50K income group has significantly higher counts across all races, with White individuals forming the largest portion, followed by Black, Asian-Pac-Islander, and other racial groups.
- In the >50K income group, the White population still dominates, though the count is much lower compared to the <=50K group.
- Other racial groups, including Black, Asian-Pac-Islander, Amer-Indian-Eskimo, and Other, have a much smaller representation in the higher-income category.
- The disparity between income levels among different races is evident, with fewer individuals from minority racial groups earning more than 50K.

This visualization highlights income distribution disparities based on race, showing that income levels vary significantly across racial groups in the dataset.

4.3 Age Distribution Plot



This histogram with a density plot illustrates the distribution of the age variable in the dataset.

Observations:

1.Right-Skewed Distribution:

- The distribution has a peak between 20 and 40 years and then gradually decreases as age increases.
- There are fewer older individuals compared to younger ones.

2.Most Common Age Group:

- The highest density is between 25 and 45 years, indicating that most individuals in the dataset belong to this age range.

3.Decline After 45:

- The density starts declining after 40-50 years, suggesting fewer individuals in older age groups.

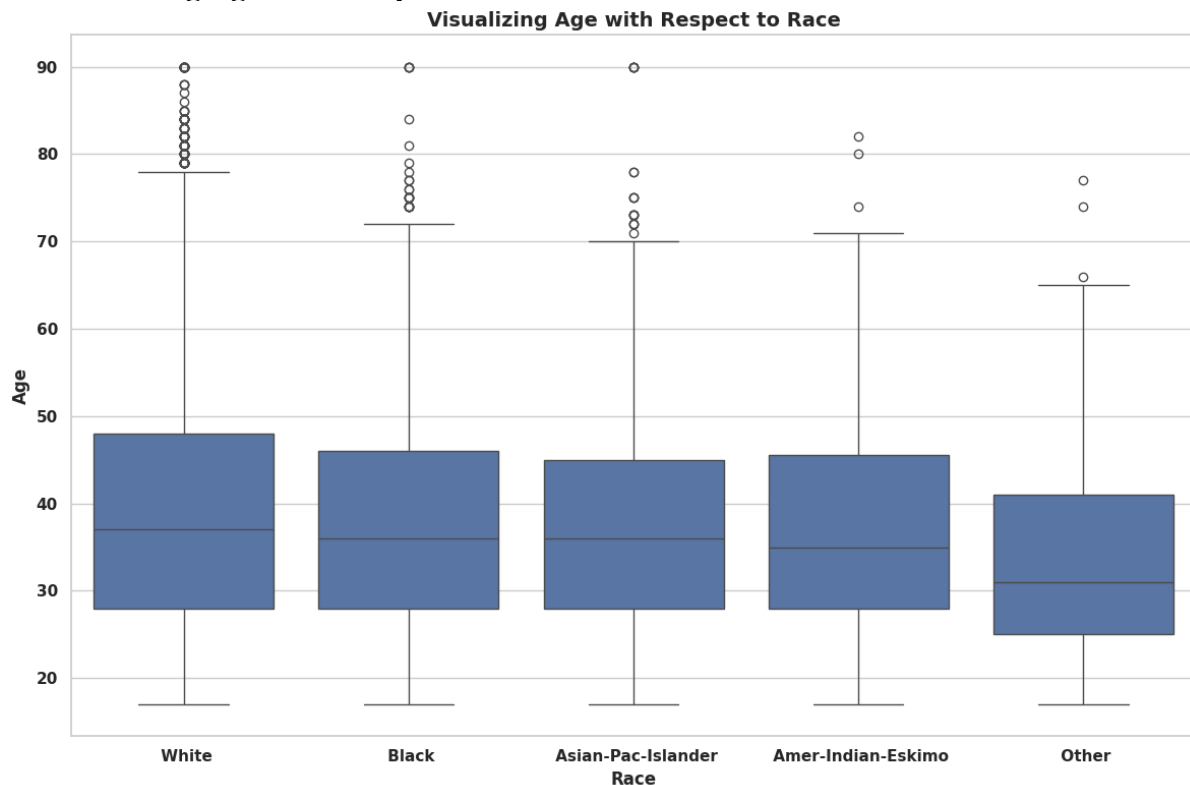
4.Long Tail Towards Older Ages:

- Some individuals are in their 60s, 70s, and even 80s, but they represent a much smaller portion of the dataset.

Conclusion:

The dataset primarily consists of younger to middle-aged individuals, with a declining number of older participants. The right-skewed nature suggests that the majority of the population is working-age, with fewer older individuals.

4.4 Visualizing Age With Respect to Race



This box plot visualizes the age distribution with respect to race.

Observations:

1. Median Age Across Races:

- The median age appears to be similar across all racial groups, generally around 35-40 years.
- The “Other” category has a slightly lower median age, suggesting a younger distribution.

2. Interquartile Range (IQR) Comparison:

- The middle 50% of individuals (IQR) in all racial groups mostly fall within 20 to 50 years.
- The “Other” group has a slightly narrower IQR, indicating a more concentrated age range.

3. Outliers:

- All groups have outliers above 70 years, showing a few elderly individuals in each category.
- The “White” group has the most outliers above 80 years, suggesting a higher proportion of older individuals.

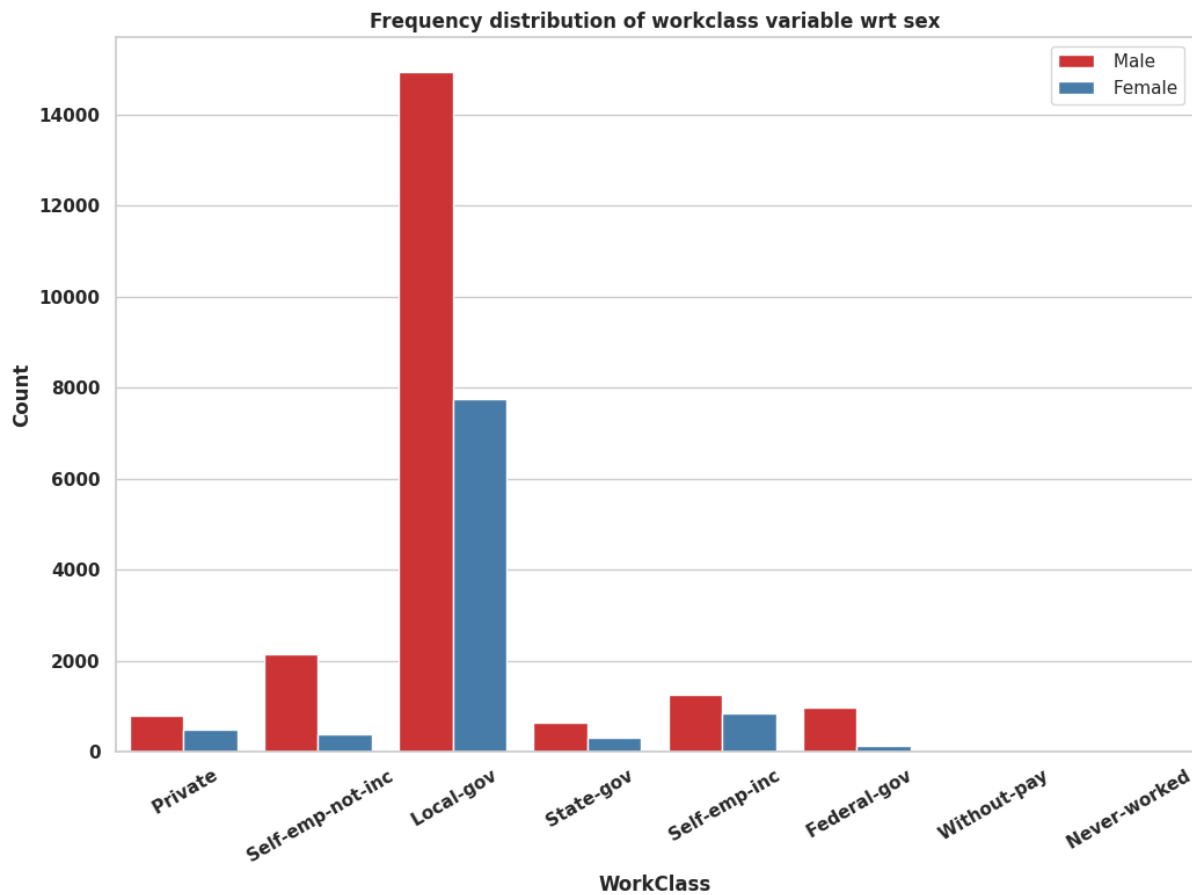
4. General Age Spread:

- The youngest individuals (~18 years) appear across all racial groups.
- The oldest individuals (~90 years) are found across most groups, but they are rare.

Conclusion:

- The age distribution across races is quite similar.
- The “Other” group tends to have younger individuals, while the “White” group has more elderly outliers.
- Overall, all races have a comparable median age and similar variability in age distribution.

5. Frequency distribution of Work class Variable with respect to Sex(Gender)



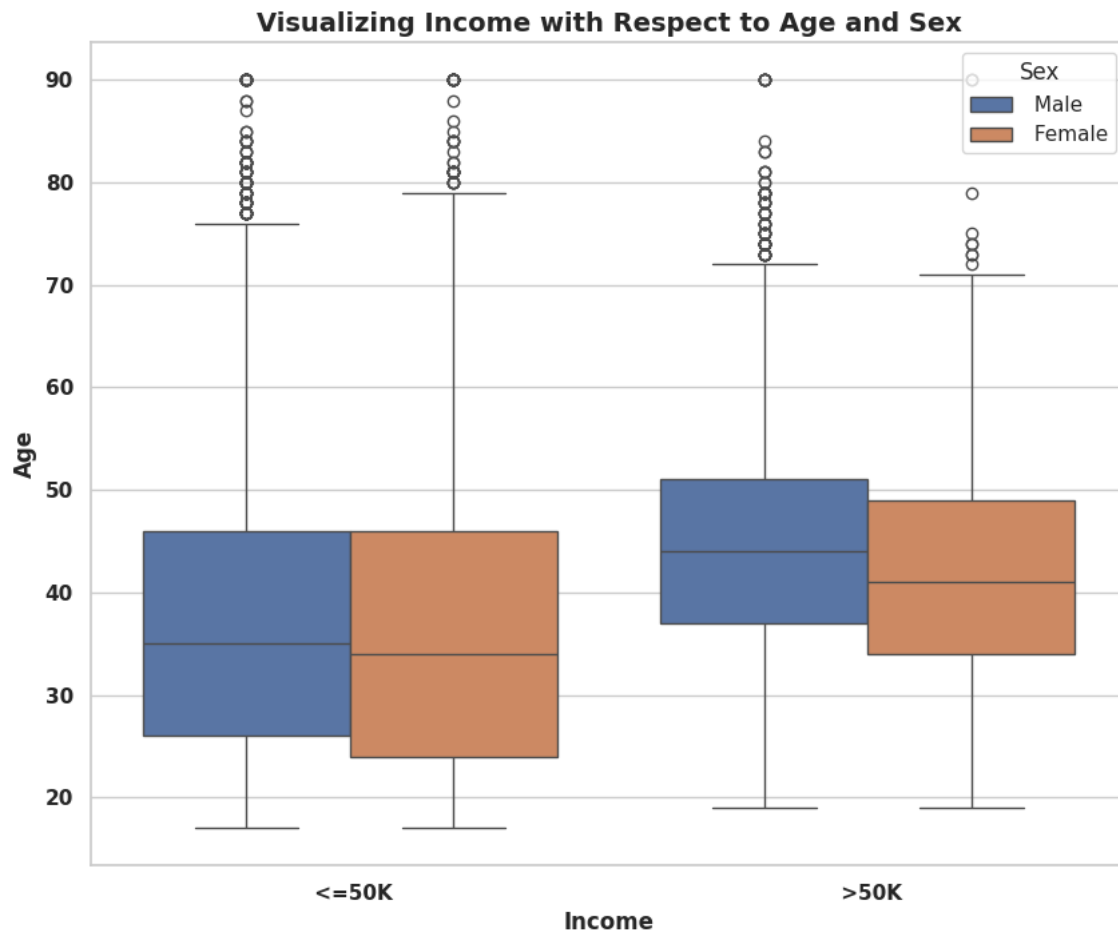
This bar chart illustrates the distribution of work class categories by gender (Male vs. Female).

Observations:

- Local government jobs have the highest representation, with significantly more males than females.
- Self-employed (both incorporated and not incorporated) roles are dominated by males, indicating a gender gap in entrepreneurship.
- Private sector employment shows a noticeable disparity, with more males than females.
- Federal and state government jobs have a relatively smaller but more balanced distribution between males and females.
- Very few individuals fall under “Without-pay” or “Never-worked” categories, with a slight male predominance.

This visualization highlights a gender disparity across work sectors, with men being more prevalent in self-employment and private jobs, while government roles show a better gender balance.

6. Visualizing Income with Respect to Age and Sex:



This box plot visualizes income distribution with respect to age and sex.

Observations:

1. Income Groups (<=50K and >50K) with Sex Comparison:

- The data is split into two income categories (<=50K and >50K).
- Within each category, we have separate distributions for males (blue) and females (brown).

2. Median Age Difference:

- In both income groups, the median age of males is slightly higher than that of females.
- For <=50K, the median age for both genders is around 30-35 years.
- For >50K, the median age for both genders is around 40-50 years.

3. Interquartile Range (IQR) Comparison:

- For <=50K, the IQR (middle 50% of the data) spans 20 to 45 years for both males and females.
- For >50K, the IQR spans 30 to 55 years for both genders, indicating that higher-income individuals tend to be older.

4. Outliers:

- In both income groups, some individuals over 70-80 years are considered outliers.
- There are slightly more male outliers (older individuals with extreme ages) in both groups.

5. Gender Differences in High Income (>50K):

- The >50K group has more males than females.
- The age distribution for females is slightly lower than males, meaning females earning >50K tend to be younger compared to their male counterparts.

Conclusion:

- Older individuals tend to have higher incomes.
- More males earn >50K compared to females, suggesting a potential gender income disparity.
- Females in the >50K group tend to be slightly younger than males in the same income category.
- Younger individuals dominate the <=50K income category, irrespective of gender.

5. Steps for training and evaluating the Random Forest Model

5.1 Prepare input data.

X_input refers to the feature set used to train the model.

Define **Y_input** as the model's target variable for testing.

5.2 Split the data:

5.3 Separate the dataset into training and testing subgroups.

- 70-80% of data is used for training, with the remaining 20-30% for testing.

5.4 Train the model.

- Initialise a **Random Forest Classifier** with the desired number of decision trees.
- Fit the model with the training dataset (X_train, Y_train).
- Evaluate model performance:
- Use trained model to forecast on test dataset (X_test).
- Analyze accuracy by comparing expected and actual values from Y_test.

6. Report on classification:

1. For this research, I developed a **Random Forest Classifier** to forecast an individual's income. I construct two models, one with ten decision trees and the other with one hundred.
2. The model accuracy score is **0.8446** for **10 decision-trees** and **0.8521** for **100 decision-trees**. Therefore, as the number of decision trees in the model increases, so does the expected accuracy.
3. Using the Random Forest model, I have identified only the most significant features, constructed the model with these features, and observed how it affects accuracy.
4. I rebuilt the model, eliminated the **native_country_41** variable, and verified its correctness. After removing the **native_country_41** variable, the model's accuracy is **0.8544**. Thus, we can observe that the removal of the **native_country_41** variable from the model has increased its accuracy.
5. An additional tool for visualizing the model's performance is the confusion matrix and classification report. They produce quality results.

7. References

1. Random Forest Algorithm in Machine Learning

Simplilearn

This tutorial provides a comprehensive overview of the Random Forest algorithm, including its working principles and applications.

[Available at: <https://www.simplilearn.com/tutorials/machine-learning-tutorial/random-forest-algorithm>]

2. Random Forest Algorithm in Machine Learning

GeeksforGeeks

An in-depth article explaining the Random Forest algorithm, its intuition, and implementation details.

[Available at: <https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/>]

3. RandomForestClassifier — scikit-learn 1.6.1 documentation

Scikit-learn

Official documentation detailing the implementation and parameters of RandomForestClassifier in Scikit-learn.

[Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>]

4. Feature Importance with Random Forests

GeeksforGeeks

This article explains how to determine the importance of features when using Random Forest models.

[Available at: <https://www.geeksforgeeks.org/feature-importance-with-random-forests/>]

5. Random Forest Hyperparameter Tuning in Python

GeeksforGeeks

A guide on optimizing Random Forest models by tuning hyperparameters using Python.

[Available at: <https://www.geeksforgeeks.org/random-forest-hyperparameter-tuning-in-python/>]

6. Difference Between Random Forest and Decision Tree

GeeksforGeeks

An article comparing Random Forest and Decision Tree algorithms, highlighting their differences and use cases.

[Available at: <https://www.geeksforgeeks.org/difference-between-random-forest-and-decision-tree/>]

7. Random Forest Classifier: Basic Principles and Applications

Serokell

An exploration of the fundamental principles of Random Forest classifiers and their practical applications.

[Available at: <https://serokell.io/blog/random-forest-classification>]

8. Random Forest - How to handle overfitting

StackExchange

A discussion on techniques to prevent overfitting in Random Forest models.

[Available at: <https://stats.stackexchange.com/questions/111968/random-forest-how-to-handle-overfitting>]

9. Random Forest Regression in Python

GeeksforGeeks

A tutorial on implementing Random Forest regression using Python, with code examples.

[Available at: <https://www.geeksforgeeks.org/random-forest-regression-in-python/>]

10. Random Forest Classification with Scikit-Learn

DataCamp

This tutorial covers how and when to use Random Forest classification with Scikit-Learn, focusing on concepts, workflow, and examples.

[Available at: <https://www.datacamp.com/tutorial/random-forests-classifier-python>]