

# Random Forest Tutorial Report

**Module:** Machine Learning and Neural Networks

**Assignment Type:** Individual Report

**Student Name:** Ajay Nunna

**Student ID:** 23096908

**Submission Date:** 27 March 2025

**Tutor:** Peter Scicluna

**GitHub link:** <https://github.com/ajaynunna/Machine-Learning-and-Neural-Networks>

# **Random Forest Tutorial Report**

## **Table of Contents**

- 1. Introduction**
  - 1.1 What is Random Forest?**
  - 1.2 Random Forest – How It Works**
  - 1.3 Visualizing Majority Voting in Random Forest**
- 2. Foundations of the Forest: Core Concepts**
  - 2.1 Bootstrapping & Feature Sampling**
  - 2.2 How Random Forest Makes Predictions**
  - 2.3 Built-in Overfitting Protection**
- 3. Building and Evaluating a Random Forest Model**
  - 3.1 Confusion Matrix Explained**
  - 3.2 Feature Importance Insight**
  - 3.3 Actual vs Predicted Distribution**
- 4. Measuring What Matters: Evaluation Metrics**
  - 4.1 Core Metrics**
  - 4.2 Visual Evaluation**
  - 4.3 Why It Matters**
- 5. Real-World Applications of Random Forest**
  - 5.1 Key Uses**
  - 5.2 Why It Works Getting It Right: Best Practices and Model Transparency**
  - 6.1 Key Best Practices**
  - 6.2 Why It Matters**
- 6. Knowing the Limits: Challenges and Limitations**
  - 7.1 Key Challenges & Solutions**
  - 7.2 Summary Table: Challenges & Solutions**
- 7. Accessibility Considerations**
- 8. Future Enhancements and Exploration**
- 9. Conclusion**
- 10. References**

# Random Forest Tutorial Report

## 1.Introduction

In machine learning, Random Forest stands out for its simplicity, power, and reliability. Instead of relying on one decision tree, it builds many each seeing a different slice of the data and combines their votes for stronger, more stable predictions.

This tutorial goes beyond code. It's about understanding how Random Forest works, why it matters, and how to apply it in Python. Whether you're a beginner or exploring ensemble methods, you'll find clear, practical steps to follow.

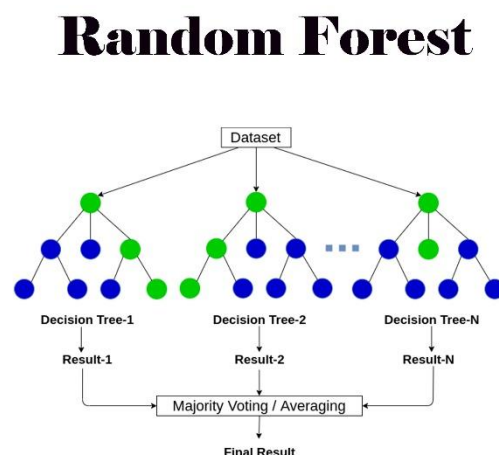
Let's dive into Random Forest a model built on many small decisions, working together for smarter results.

### 1.1 What is Random Forest?

Random Forest is a supervised machine learning algorithm that combines multiple decision trees to produce a more accurate and robust prediction.

### 1.2 Random Forest – How It Works

This diagram clearly illustrates the Random Forest workflow. The dataset is split across multiple decision trees, each learning different patterns. Once trained, every tree gives its own prediction.



Random Forest combines multiple trees using **majority voting** or **averaging**, making predictions more accurate, stable, and resistant to overfitting.

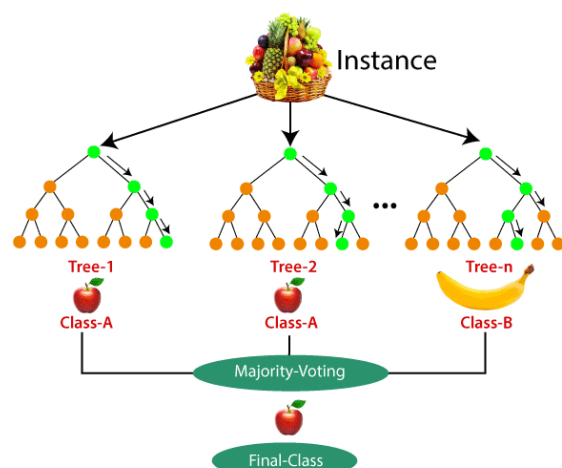
It's teamwork simple models joining forces for stronger results.

### 1.3 Visualizing Majority Voting in Random Forest

This image offers a fun and intuitive way to understand how Random Forest makes decisions. Imagine trying to identify the type of fruit in a basket. Instead of relying on one guess, we ask multiple decision trees—each trained slightly differently.

- **Tree-1** and **Tree-2** say it's an **Apple (Class-A)**.
- **Tree-n** says it's a **Banana (Class-B)**.

Random Forest doesn't choose the loudest voice it chooses the **majority**. Since most trees voted for Apple, that becomes the **final prediction**. This "group decision" process is what makes Random Forest so reliable. It's not about being perfect individually but **being powerful together**.



# Random Forest Tutorial Report

## 2. Foundations of the Forest: Core Concepts

Before jumping into code, it's key to understand how Random Forest works. It's more than just multiple decision trees—it's smart use of **randomness** and **teamwork** to boost performance.

### 2.1 Bootstrapping & Feature Sampling

#### Bootstrapping (Bagging):

Each tree is trained on a different random sample of the dataset (with replacement). Some rows repeat, others are left out adding variety.

#### Feature Sampling:

At each split, only a random subset of features is considered.

- Classification: uses  $\sqrt{M}$  features
- Regression: uses  $M/3$  features

This randomness ensures diversity among trees, reducing overfitting and improving stability.

### 2.2 How Random Forest Makes Predictions

Once the forest is grown, it's time to make predictions. But instead of asking just one tree, we ask them all, and combine their responses.

#### For Classification

Each tree makes a prediction, and the class with the most votes becomes the final answer.

$$\hat{y} = \text{mode}(y_1, y_2, \dots, y_n)$$

Where:

$y_1, y_2, \dots, y_n$  are the predictions from each of the  $n$  trees

#### For Regression

Each tree outputs a number, and the average of all those numbers becomes the final prediction.

$$\hat{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

This aggregation majority voting or averaging makes Random Forest more reliable. One tree might get it wrong, but the group is usually right.

### 2.3 Built-in Overfitting Protection

Unlike single trees, Random Forest resists overfitting by design:

- Diverse Trees: Each tree sees different data and features.
- Ensemble Averaging: Combines outputs to reduce individual errors.
- OOB Validation: Uses leftover data for built-in accuracy checks.

To further improve generalization:

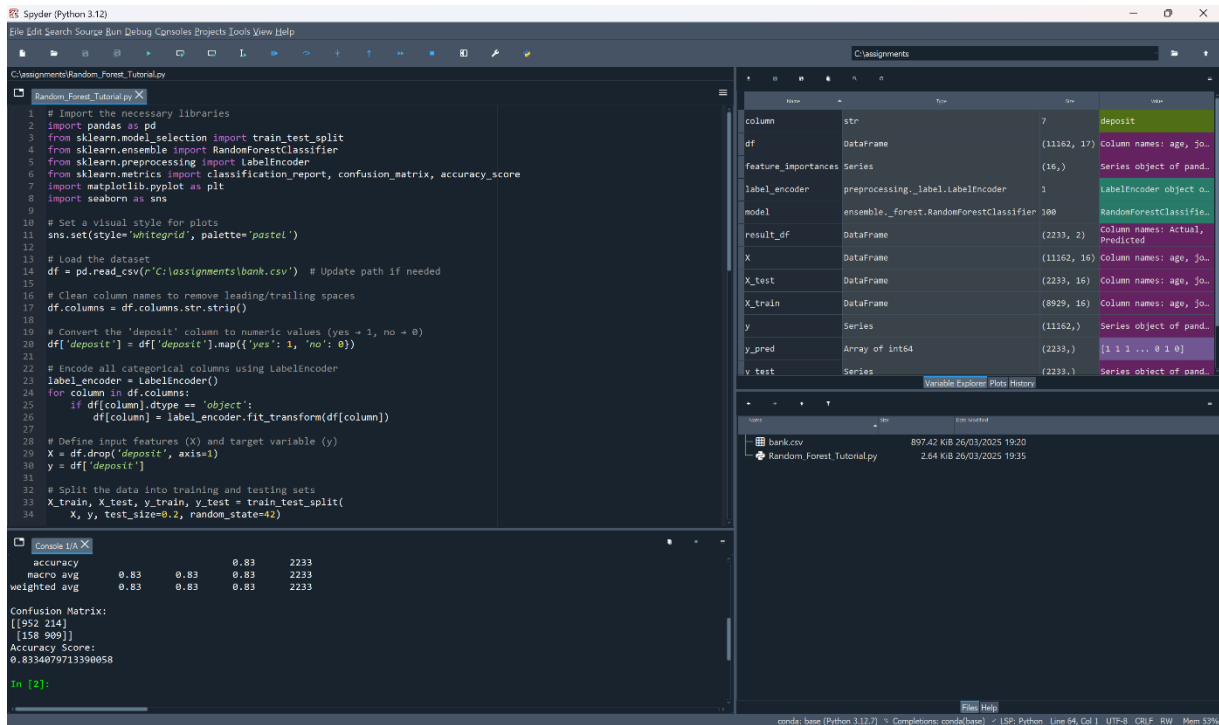
- Limit tree depth
- Set minimum samples per split
- Restrict features per split

Random Forest thrives by blending randomness with structure—many trees, one strong decision.

# Random Forest Tutorial Report

## 3. Building and Evaluating a Random Forest Model

Trains a Random Forest on the Bank Marketing dataset—covers data prep, training, evaluation, and visual insights like confusion matrix and feature importance. A complete, end-to-end workflow.



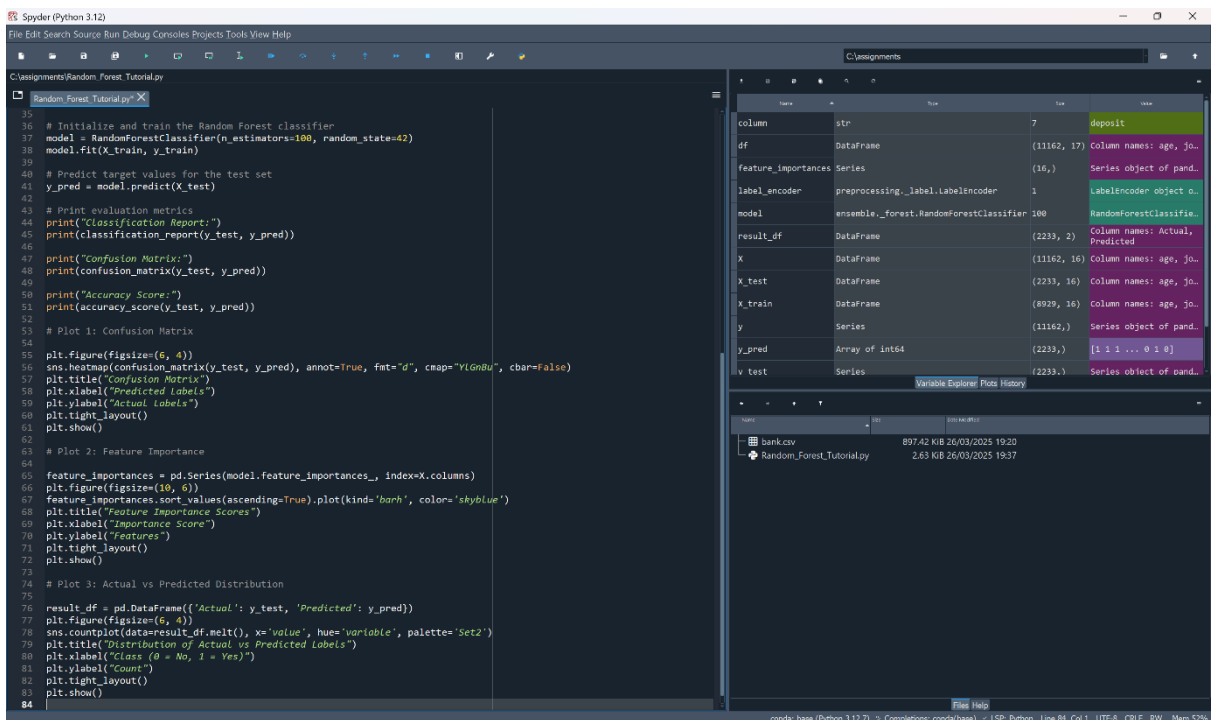
```
1 # Import the necessary libraries
2 import pandas as pd
3 from sklearn.model_selection import train_test_split
4 from sklearn.ensemble import RandomForestClassifier
5 from sklearn.preprocessing import LabelEncoder
6 from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
7 import matplotlib.pyplot as plt
8 import seaborn as sns
9
10 # Set a visual style for plots
11 sns.set(style='whitegrid', palette='pastel')
12
13 # Load the dataset
14 df = pd.read_csv(r"C:\assignments\bank.csv") # Update path if needed
15
16 # Clean column names to remove leading/trailing spaces
17 df.columns = df.columns.str.strip()
18
19 # Convert the 'deposit' column to numeric values (yes = 1, no = 0)
20 df['deposit'] = df['deposit'].map({'yes': 1, 'no': 0})
21
22 # Encode all categorical columns using LabelEncoder
23 label_encoder = LabelEncoder()
24 for column in df.columns:
25     if df[column].dtype == 'object':
26         df[column] = label_encoder.fit_transform(df[column])
27
28 # Define input features (X) and target variable (y)
29 X = df.drop('deposit', axis=1)
30 y = df['deposit']
31
32 # Split the data into training and testing sets
33 X_train, X_test, y_train, y_test = train_test_split(
34     X, y, test_size=0.2, random_state=42)
```

Console I/O:

```
accuracy      0.83      0.83      0.83      2233
macro avg     0.83      0.83      0.83      2233
weighted avg  0.83      0.83      0.83      2233

Confusion Matrix:
[[952 214]
 [158 909]]
Accuracy Score:
0.8334079713390058

In [2]:
```



```
35
36 # Initialize and train the Random Forest classifier
37 model = RandomForestClassifier(n_estimators=100, random_state=42)
38 model.fit(X_train, y_train)
39
40 # Predict target values for the test set
41 y_pred = model.predict(X_test)
42
43 # Print evaluation metrics
44 print('Classification Report:')
45 print(classification_report(y_test, y_pred))
46
47 print('Confusion Matrix:')
48 print(confusion_matrix(y_test, y_pred))
49
50 print('Accuracy Score:')
51 print(accuracy_score(y_test, y_pred))
52
53 # Plot 1: Confusion Matrix
54
55 plt.figure(figsize=(6, 4))
56 sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='YlGnBu', cbar=False)
57 plt.title('Confusion Matrix')
58 plt.xlabel('Predicted Labels')
59 plt.ylabel('Actual Labels')
60 plt.tight_layout()
61 plt.show()
62
63 # Plot 2: Feature Importance
64
65 feature_importances = pd.Series(model.feature_importances_, index=X.columns)
66 plt.figure(figsize=(10, 6))
67 feature_importances.sort_values(ascending=True).plot(kind='barh', color='skyblue')
68 plt.title('Feature Importance Scores')
69 plt.xlabel('Importance Score')
70 plt.ylabel('Features')
71 plt.tight_layout()
72 plt.show()
73
74 # Plot 3: Actual vs Predicted Distribution
75
76 result_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
77 plt.figure(figsize=(6, 4))
78 sns.countplot(data=result_df.melt(), x='value', hue='variable', palette='Set2')
79 plt.title('Distribution of Actual vs Predicted Labels')
80 plt.xlabel('Class (0 = No, 1 = Yes)')
81 plt.ylabel('Count')
82 plt.tight_layout()
83 plt.show()
84
```

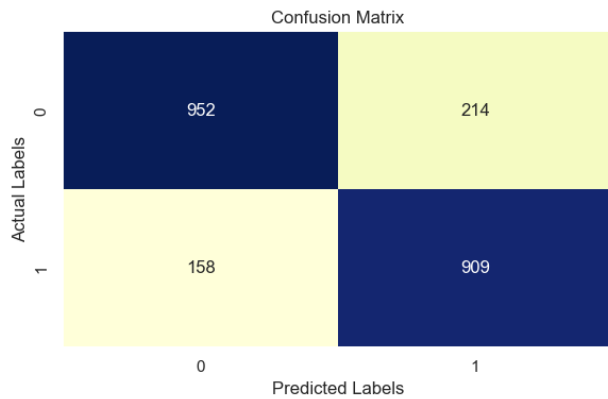
The visual outputs offer a clear, intuitive view of model performance highlighting classification accuracy, feature importance, and how well predictions are balanced across classes.

# Random Forest Tutorial Report

## 3.1 Confusion Matrix Explained

The confusion matrix shows model performance:

- 952 true negatives (correct non-subscriber predictions)
- 909 true positives (correct subscriber predictions)
- 214 false positives (predicted subscription, but not)
- 158 false negatives (missed subscribers)

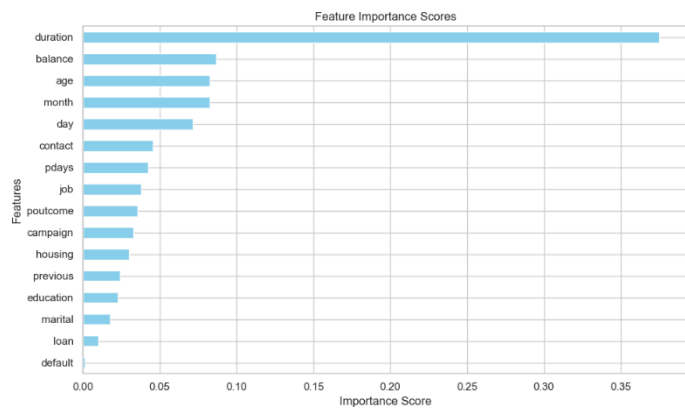


The model performs well, with room for improvement in false positives and negatives.

## 3.2 Feature Importance Insight

This plot shows which features had the most influence on the model's predictions.

- Duration stands out as the most important factor longer calls strongly indicate subscription likelihood.
- Balance, age, and month also contributed meaningfully to the decision-making.
- Features like loan, marital status, and default had minimal impact.

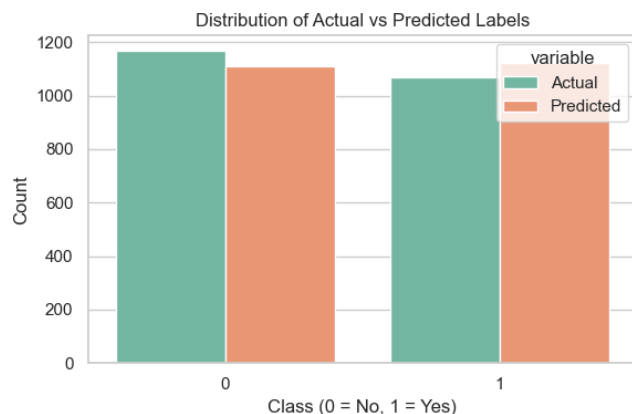


Understanding feature importance helps us interpret the model and focus on the variables that truly drive outcomes.

## 3.3 Actual vs Predicted Distribution

This bar chart compares the number of actual and predicted labels for each class.

- The bars are closely aligned, showing that the model predicts both classes with balanced accuracy.
- Slight differences hint at minor misclassifications, but overall, the prediction pattern mirrors the actual class distribution well.



This visual confirms that the model is not biased toward one class and maintains consistency across outputs.

This code not only builds a reliable Random Forest model but also walks us through how to evaluate it effectively. From training and prediction to visual interpretation, each step adds clarity to the model's performance and behavior. These insights set the stage for deeper evaluation and meaningful conclusions in the next sections.

# Random Forest Tutorial Report

## 4. Measuring What Matters: Evaluation Metrics

Training a model is just half the story; what truly matters is its performance on new data. This section evaluates our Random Forest classifier using key metrics and visual tools for a full understanding of its accuracy and decision-making confidence.

### 4.1 Core Metrics

- **Accuracy:** Measures overall correctness but can be misleading in imbalanced datasets.
- **Precision:** Focuses on how many predicted subscribers actually did subscribe.
- **Recall:** Captures how well the model identifies true subscribers.
- **F1 Score:** Balances precision and recall for a more comprehensive view.

### 4.2 Visual Evaluation

- **Confusion Matrix:** Breaks down correct vs incorrect predictions, highlighting strengths and weaknesses.
- **Feature Importance:** Ranks features by their influence on predictions (e.g., duration was the most important).
- **Actual vs Predicted Distribution:** Confirms if the model is balanced across both classes.

### 4.3 Why It Matters

Relying on a single metric can be misleading. Using a combination of metrics and visuals helps:

- Validate the model's reliability
- Spot imbalances or misclassifications
- Identify which features drive predictions
- Build confidence in real-world applications

This comprehensive evaluation ensures performance and builds trust in the model's decisions, crucial when they impact people, products, or business outcomes.

## 5. Real – World Applications of Random Forest

Random Forest isn't just a theoretical concept—it's a real-world tool used across industries for making data-driven decisions. Known for its flexibility and robustness, it's ideal for complex datasets where both accuracy and transparency matter.

### 5.1 Key Uses:

- **Healthcare:** Predicts disease risks, identifies at-risk patients, and assists in diagnosis using health data.
- **Finance:** Detects fraud, assesses credit, and automates loan approvals by analysing financial behaviour.
- **Retail & Marketing:** Helps predict churn, understand customer preferences, and tailor marketing strategies.
- **Environmental & Agricultural Systems:** Predicts crop yields, monitors pollution, and supports environmental decisions.
- **Education & Behavioural Analysis:** Analyses performance trends and predicts student dropout rates.

### 5.2 Why It Works:

Random Forest efficiently handles diverse, noisy data while providing clear and reliable results with minimal setup.

# Random Forest Tutorial Report

## 6. Getting It Right: Best Practices and Model Transparency

Building a **Random Forest** model is one thing; creating one that's reliable, efficient, and interpretable is another. This section highlights strategies to fine-tune the model for consistency while making it understandable for stakeholders.

### 6.1 Key Best Practices

#### 1. Tune Hyperparameters:

Adjusting settings like `n_estimators`, `max_depth`, `min_samples_split`, and `max_features` can greatly improve performance. Use tools like **GridSearchCV** or **RandomizedSearchCV** for optimization.

#### 2. Balance Simplicity & Performance:

Avoid overly complex trees. Simpler, well-tuned models with slight pruning perform better and generalize well.

#### 3. Understand the Model:

- **Feature Importance:** Check which variables influence predictions.
- **Partial Dependence Plots:** See how changes in a feature affect outcome.

#### 4. Cross-Validate for Reliability:

Use **k-fold cross-validation** to ensure consistent performance and avoid overfitting to one data split.

#### 5. Document Your Work:

Log dataset versions, preprocessing steps, random seeds, and hyperparameters for transparency and reproducibility.

#### 6. Keep It Human-Centric:

Consider the impact on real-world decisions, ensuring fairness and transparency—especially when explaining results to non-technical stakeholders.

### 6.2 Why It Matters:

Random Forest is a powerful tool, but these best practices ensure it's used responsibly, producing accurate, trustworthy models.



# Random Forest Tutorial Report

## 7. Knowing the Limits: Challenges and Limitations

While **Random Forest** is powerful, it has its challenges. Understanding these trade-offs is crucial when applying it to real-world problems.

### 7.1 Key Challenges & Solutions:

#### 1. Overfitting on Noisy Data:

Random Forest can still be overfit with small or noisy datasets.

**Solution:** Limit tree depth, increase samples per leaf, and cross-validate.

#### 2. Slow Training on Large Datasets:

Training many trees can be computationally expensive.

**Solution:** Reduce the number of trees, parallelize training, or use **ExtraTreesClassifier** for speed.

#### 3. Lower Interpretability:

Random Forest lacks clear decision paths like single trees.

**Solution:** Use feature importance plots or **SHAP** values for better interpretability.

#### 4. Sensitivity to Imbalanced Data:

The model may bias toward the majority class.

**Solution:** Apply class weighting, resampling, or **SMOTE** to balance the data.

### 7.2 Summary Table: Challenges & Solutions

| Challenge                   | Why It Matters                                | How to Handle It  |
|-----------------------------|---|---|
| Overfitting                 | Can lead to poor generalization               | Limit depth, increase samples per split, cross-validate |
| Slow training               | Affects scalability with big data             | Reduce trees, parallelize, try ExtraTrees               |
| Reduced interpretability    | Hard to explain predictions in business terms | Use feature importance, SHAP, or reduce complexity      |
| Class imbalance sensitivity | May ignore minority class                     | Use class weights or balance the dataset                |

Recognizing these limitations allows you to use Random Forest more thoughtfully maximizing its strengths while staying aware of potential pitfalls.

# Random Forest Tutorial Report

## 8.Future Enhancements and Exploration

While the Random Forest model performs well, improvements can be made:

- **Hyperparameter Tuning:** Use **GridSearchCV** or **Bayesian optimization**.
- **Advanced Evaluation:** Add **cross-validation**, **ROC curves**, and precision/recall trade-offs.
- **Model Deployment:** Deploy via **Streamlit**, **Flask**, or as an **API**.
- **Versioning:** Track with **Git** and **DVC**, and document dependencies.
- **Explore Alternatives:** Compare with **XGBoost**, **Logistic Regression**, or **Neural Networks**.
- **Enhance Tutorial:** Add **interactive visuals**, **real-time predictions**, and **code walkthroughs**.

This project lays a solid foundation with clear paths for future growth.

## 9. Conclusion

This tutorial has walked through the full journey of building, understanding, and evaluating a Random Forest model in a real-world context. More than just running code, it focused on shaping a learning experience that connects technical concepts with practical insights.

Through the Bank Marketing dataset, we explored how Random Forest handles structured data, makes meaningful predictions, and offers interpretability through feature importance and clear evaluation metrics. Alongside accuracy, we emphasized readability, accessibility, and responsible modelling practices.

While this may conclude the current task, it's far from the end. This project now stands as a springboard for deeper exploration, real-world application, or continued learning. With a solid understanding in place, the path ahead is open, and the next step is yours to define.

# Random Forest Tutorial Report

## 10. References

- **Breiman, L.** (2001). *Random Forests*. Machine Learning, 45(1), 5–32.  
<https://doi.org/10.1023/A:1010933404324>  
The foundational research paper introducing the Random Forest algorithm.
- **Dua, D., & Graff, C.** (2019). *UCI Machine Learning Repository: Bank Marketing Dataset*. University of California, Irvine. Retrieved from  
<https://archive.ics.uci.edu/ml/datasets/Bank+Marketing>  
Source of the dataset used in this tutorial.
- **Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E.** (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.  
Reference for the Python machine learning library used to build and evaluate the model.
- **Scikit-learn Documentation.** (2023). *RandomForestClassifier*. Retrieved from  
<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>  
Official documentation for the Random Forest classifier implementation.
- **Scikit-learn Documentation.** (2023). *Model Evaluation: Classification Report & Confusion Matrix*. Retrieved from [https://scikit-learn.org/stable/modules/model\\_evaluation.html](https://scikit-learn.org/stable/modules/model_evaluation.html)  
Resource used for understanding evaluation metrics like precision, recall, and F1-score.
- **Molnar, C.** (2022). *Interpretable Machine Learning*. Retrieved from  
<https://christophm.github.io/interpretable-ml-book/>  
For general understanding of feature importance and model interpretability concepts.
- **KDNuggets** (2020). *Decision Tree Algorithm, Explained*. Retrieved from  
<https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html>  
Source for Figure 1 in the tutorial.
- **Analytics Vidhya** (2021). *Understanding Random Forest*. Retrieved from  
<https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>  
Source for Figure 1.3 in the tutorial.
- **Jain, A.** (2020). *Everything About Random Forest*. Medium. Retrieved from  
<https://medium.com/@abhishekjainindore24/everything-about-random-forest-90c106d63989>  
Source for Figure 1.2 in the tutorial.