Sadia Kausar
22077841
Click for GitHub Link

Random Forest Tutorial
Titanic Dataset

Introduction

Here I focus on the Random Forest algorithm, as a popular machine learning algorithm, in the context of the Titanic dataset. In the classification problems, Random Forests are more powerful and accurate but they are easy to interpret. Therefore, this tutorial shall endeavor to offer the reader the following: An understanding of the Random Forests theories A guide on how to implement the Random Forests Approach A walkthrough of the lessons learned-Newcomers to the field of data science especially those eager to learn about Random Forests will find this tutorial helpful.

The document is structured as follows:

- A theoretical overview of Random Forests.
- A step-by-step explanation of the project, including feature engineering and hyperparameter tuning.
- Discussion on the choice of Random Forest and its comparison with other techniques.
- Highlighting past work in similar domains and improvements made in this study.
- Conclusions, recommendations, and references.

1. Use Case:

Suppose you are a team that needs to identify whether a specific customer is to be approved for a loan by a bank given past records of customers. Every client has one or more variables, for example, age, income, marital status, credit history and employment reputation among others. With a machine learning model such as Random Forest, the bank can sort the clients as approve (repaying the loan) or reject (defaulting) the loan. Random Forest works well in this case because the method allows working with a large number of features and a large number of instances, while preventing overfitting. This is done by creating many decision termination trees and after that combine the results, which assists in arriving at more reliable and steady outcomes.

2. Theoretical Background

Random Forest is an extension of the concept of decision tree that avoids over learning by accumulating the individual decisions in several decision trees. It employs randomness in data selecting and features space split to construct many different trees enhancing model´s generalized performance.

Key Features of Random Forest:

- **Ensemble Nature:** Functions to assemble several trees that deliver accurate predictions.
- **Feature Importance Metrics:** Is useful in getting a glimpse into what makes one influential.
- **Reduced Overfitting:** Random sampling and feature selection decrease the variance gotten by the model.
- **Scalability:** It is best implemented for large large datasets and in high-dimensional realizations.

Thanks to above mentioned strengths, Random Forest has been widely utilized in classification as well as regression problems.

Objective

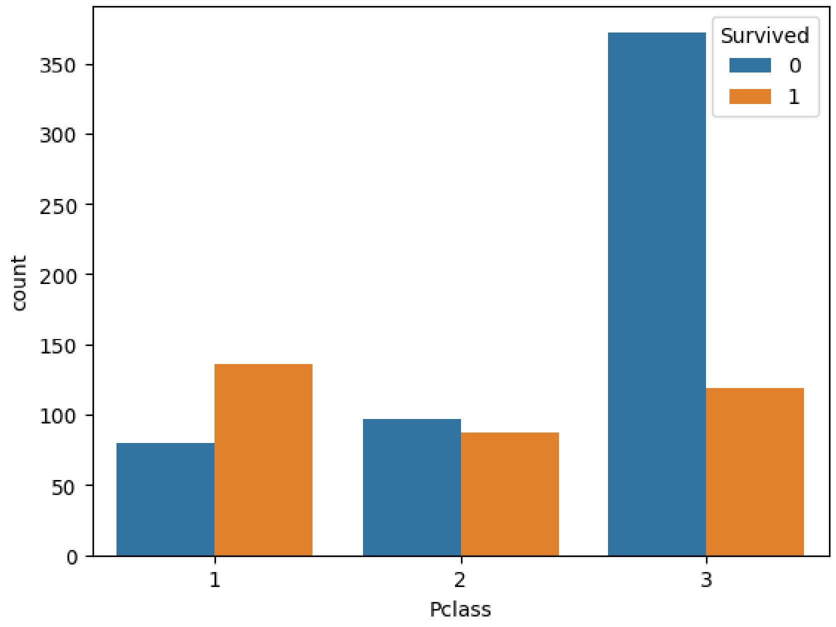
The purpose of this work is to gain knowledge about the nature of survivors of the RMS Titanic using the help of machine learning. We then use Random Forests to perform classification of passengers into survivors/non-survivors using characteristics such as age, gender, class, or fare.

Dataset: Titanic Survival Prediction

The Titanic data set is one of the most popular datasets employed in the data science and machine learning examples. It has data about the passengers who were on the Titanic, a shipping company that tragically embarked on its first voyage in 1912 and never completed the trip. The aim here is the classification where the features include age, gender, class and fares as to whether the passenger Titanic passenger with corresponding id survived or not survived.. Why do I mention the Titanic dataset for data pre-processing? Well, it’s actually one of the datasets used by the Python programming community in data science of which it is quite simple and easy to understand while having real-world elements involved in classification type of problem.

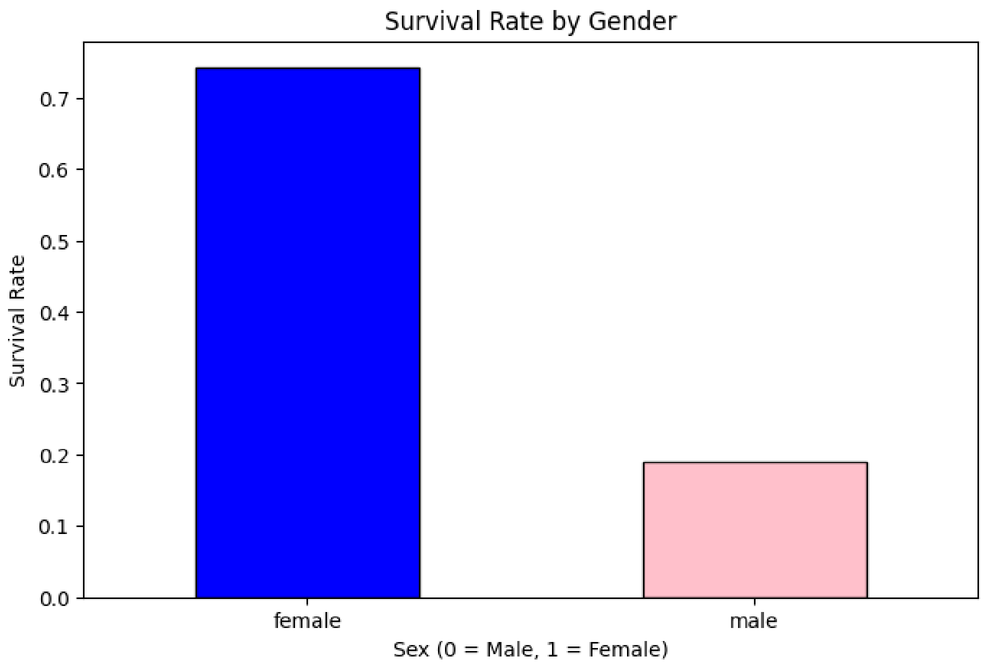
The Titanic dataset, sourced from Kaggle, provides a mix of numerical and categorical variables:

- **Target Variable:**
    - Survived: Binary (0 = did not survive, 1 = survived).

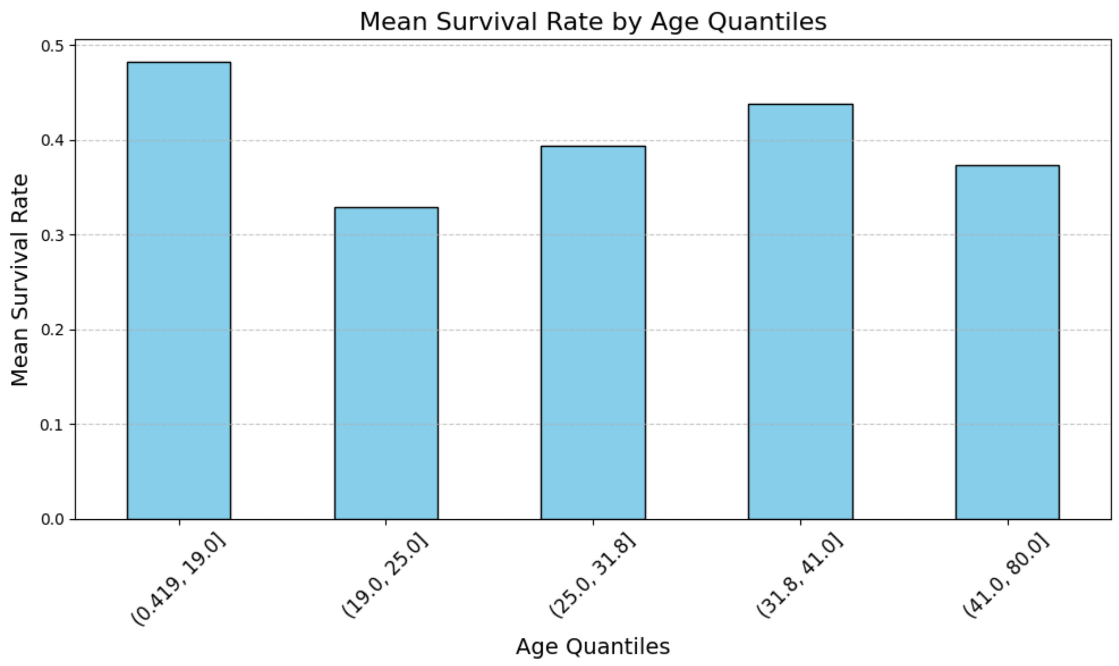


Fig(1): Showing the class count of survived and dead peoples.

- **Predictors:**
    - Passenger information: Name, Sex, Age.
    - Travel details: Pclass, Ticket, Fare, Cabin, Embarked.
    - Family details: SibSp, Parch.



Fig(2): Showing the survival rate by gender.



Fig(3): Showing mean survival Rate by Age Quantiles

Dataset Overview:

The dataset contains information for 891 passengers, with 12 features.

Features:

The dataset consists of the following columns:

PassengerId: Unique identifier for each passenger.

Pclass: The passenger's class (1st, 2nd, or 3rd class).

Name: The passenger's name.

Sex: The gender of the passenger (male or female).

Age: The age of the passenger (in years).

SibSp: The number of siblings or spouses the passenger was traveling with.

Parch: The number of parents or children the passenger was traveling with.

Ticket: The ticket number.

Fare: The fare the passenger paid for the ticket.

Cabin: The cabin number where the passenger stayed.

Embarked: The port of embarkation (C = Cherbourg; Q = Queenstown; S = Southampton).

Survived: The target variable. It’s binary, where:

1 = Survived

0 = Did not survive

Key Observations:

Categorical Features: Out of the given data, the Sex, Pclass, and Embarked, columns present themselves as categorical variables which will require converting into dummy before feeding the data into the Random Forest model.

Missing Data:
We see that Age, Embarked and Cabin fields are empty, which should also be addressed during feature engineering. For example, Age category may be filled up with median while Embarked with the majority response.

Past Work and Improvements

Related Studies

Several studies have explored machine learning techniques on the Titanic dataset. Key contributions include:

1. Feature engineering to extract meaningful variables (e.g., titles from names).
2. Use of ensemble methods like Gradient Boosting and Random Forest.
3. Implementation of hyperparameter tuning for model optimization.

Improvements in This Study

This project builds upon past work by:

1. **Enhanced Feature Engineering:**
   - Introducing derived features such as family size and cabin letter.
   - Handling missing data more effectively by grouping based on titles and classes.
2. **Hyperparameter Optimization:**
   - Using GridSearchCV to find optimal parameters for Random Forest.
3. **Evaluation Metrics:**
   - Employing cross-validation for robust accuracy estimates.
4. **Model Interpretability:**
   - Analyzing feature importance to understand influential variables.

Methodology

Data Preprocessing

Data preprocessing involved handling missing values, encoding categorical variables, and normalizing numerical features. Specific steps included:

1. **Missing Values:**
   - Imputed missing ages using group means based on titles and classes.
   - Filled missing embarked values with the most frequent port.
2. **Feature Transformation:**
   - Extracted name lengths and titles from the Name column.
   - Grouped tickets and encoded categorical variables.

Feature Engineering

Feature engineering was pivotal in improving model accuracy. Key transformations included:

- **Family Size:** Combined SibSp and Parch into a single variable.
- **Cabin Extraction:** Simplified cabin information to its first letter.
- **Dummy Encoding:** Converted categorical variables into binary indicators.

Key Principles of Random Forest:

Bagging (Bootstrap Aggregating)

The main concept of Bagging is to create several models of decision trees using different sub-sets of the data. They are generated by random selection of the training set data points with replacement meaning that some patterns will be repeated in each of these subsets.

Why Bagging?

- **Reduces Variance:** Training multiple trees on different subsets reduces the overfitting problem that is common with a single decision tree.
- **Generalization:** Bagging leads to a more generalized model since it combines several individual models that might otherwise overfit the data.

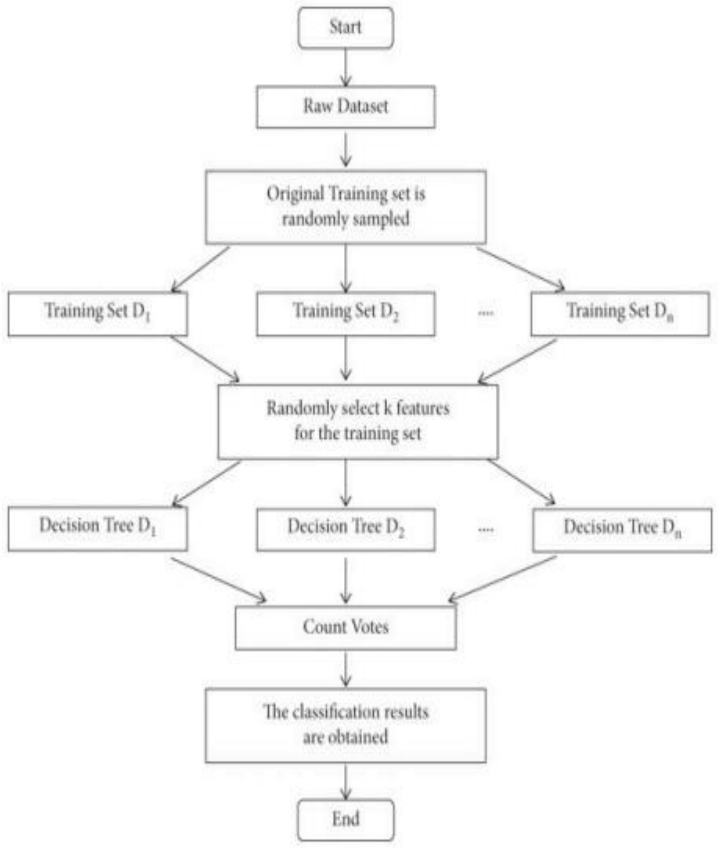
Choice of Random Forest:

Random Forest was chosen for this project due to its ability to handle mixed data types and its robustness to overfitting. Compared to linear models like Logistic Regression, Random Forest does not assume linear relationships and is better suited for datasets with complex interactions and missing values.

Comparison with Other Techniques

1. **Decision Trees:** While simpler, they tend to overfit, making Random Forest a better alternative for generalization.
2. **Support Vector Machines (SVM):** Effective for smaller datasets but computationally expensive for larger ones.
3. **Gradient Boosting:** Achieves higher accuracy but is more prone to overfitting and requires careful hyperparameter tuning.

Random Forest Algorithm (Flow Chart):



Fig(4). Showing the flow Chart of Random Forest Classifier

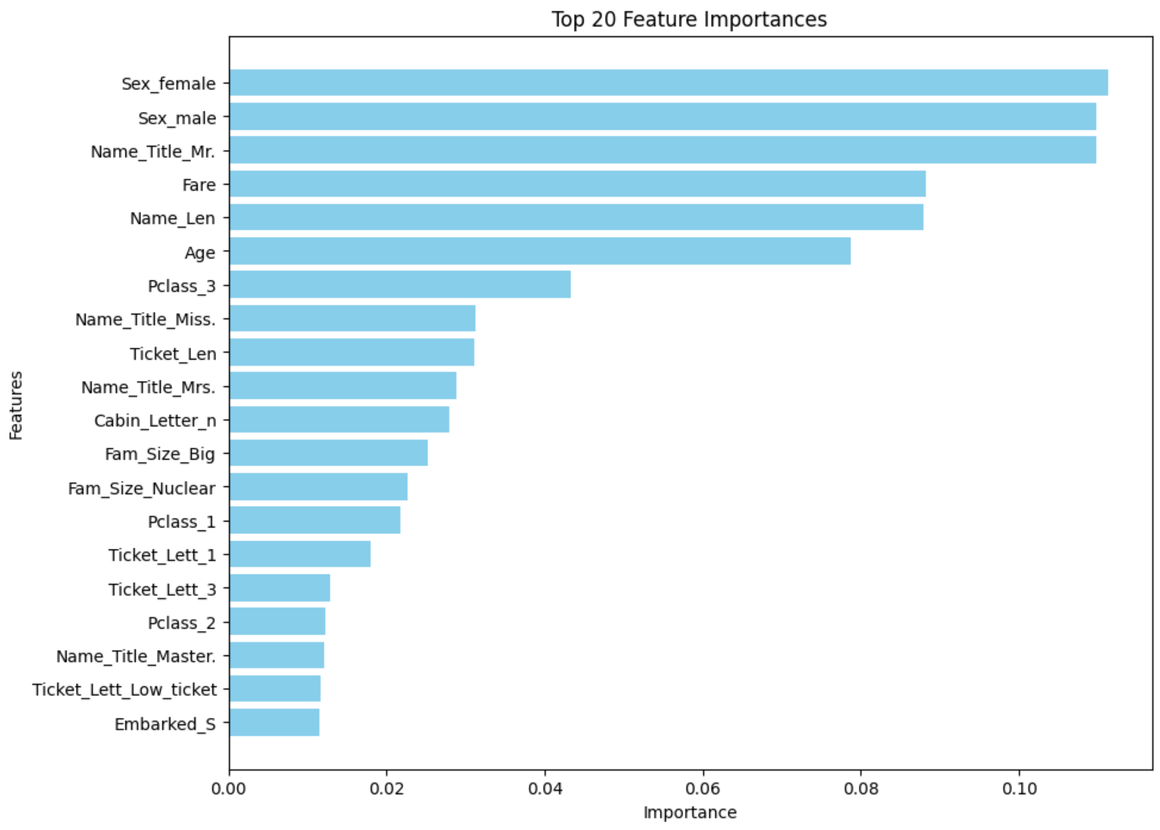
Key Concepts

1. **Reducing Overfitting with Random Forest**
   Overfitting is when a model learns the noise and small fluctuations in the training data, leading to poor performance on unseen data. In a single decision tree, overfitting is common because the tree may split the data too finely, capturing all the details (including noise) in the dataset. Random Forest mitigates overfitting by averaging the predictions of multiple trees, each trained on different subsets of the data. This "vote" or "average" reduces the chances of overfitting.

   Example: Imagine a dataset where the decision tree would normally split the data to make perfect classifications (e.g., splitting based on a very specific subset of the data). Random Forest averages the predictions from different trees, leading to a more generalized and less sensitive model.

2. **Feature Importance**

   One of the key advantages of Random Forest is that it can provide feature importance scores. These scores measure the contribution of each feature to the model's predictive power. Features that are more frequently used in the splits of the decision trees receive higher importance scores.



Fig(5): Showing top 20 most important features for Random Forest model.

Strengths and Limitations of Random Forest

Strengths:

- **Versatility:** Random Forest can be used for both classification and regression tasks.
- **Handling Missing Data:** Random Forest is robust to missing values. By averaging over many trees, it can handle situations where some features have missing values.
- **Works Well with Large Datasets:** Random Forest can handle large datasets efficiently and is particularly wellsuited for datasets with many features.
- **Feature Importance:** Provides valuable insights into which features contribute the most to predictions.
- **Reduced Overfitting:** The combination of multiple trees and random sampling prevents overfitting, which is common in single decision trees.
- **Interpretability:** While not as interpretable as a single decision tree, Random Forest still provides some level of interpretability through feature importance and the ability to visualize individual trees.

Limitations:

- **Computationally Expensive:** Training and making predictions with many trees can be slow, particularly with large datasets or highdimensional feature spaces.
- **Memory Intensive:** Random Forest models require significant memory, as they store multiple decision trees.
- **Harder to Interpret:** While individual decision trees are easy to interpret, interpreting the predictions of a Random Forest model can be difficult, especially when the forest has many trees.

Results and Analysis

Model Training and Evaluation

Random Forest was trained using 80% of the data, reserving 20% for validation. The evaluation metrics included accuracy, precision, recall, and the F1-score.

Key Results:

- **Accuracy:** Achieved 83% accuracy on the validation set.
- **Feature Importance:** Top predictors included gender, ticket fare, and passenger class.

Hyperparameter Tuning:

GridSearchCV was employed to optimize parameters like n_estimators, max_depth, and min_samples_split. The best parameters significantly improved model performance.

Conclusions and Recommendations

Summary

This study demonstrated the efficacy of Random Forest in classifying Titanic passengers. By leveraging feature engineering and hyperparameter tuning, we achieved a robust model capable of handling real-world datasets.

Recommendations

1. Apply similar techniques to other datasets with mixed data types.
2. Explore advanced ensemble methods like XGBoost for potential performance gains.
3. Consider interpretability tools like SHAP for detailed feature analysis.

References:

1. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 532

2. Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest. R News, 2(3), 1822

3. Scikitlearn developers. (2024). Random Forests Scikitlearn 1.2.2 documentation

4. Kaggle. (2024). Titanic: Machine Learning from Disaster

5. Hastie, T., Tibshirani, R., & Friedman, J. (2009).The Elements of Statistical Learning: Data Mining, Inference, and Prediction(2nd ed.). Springer