## Documentation: Titanic Dataset Classification Models

### 1. Introduction

This documentation outlines the steps involved in building and evaluating multiple machine learning models to predict the survival of passengers on the Titanic. The analysis is based on a preprocessed version of the Titanic dataset, utilizing various preprocessing techniques and classification algoritahms.

### 2. Dependencies

The following Python packages are required:

* **numpy**: For numerical operations.
* **pandas**: For data manipulation and analysis.
* **scikit-learn**: For machine learning model implementation, preprocessing, and evaluation.
* **feature\_engine**: For advanced feature engineering tasks like encoding and outlier treatment.
* **matplotlib** and **seaborn**: For data visualization.

Install the dependencies using pip:

pip install numpy pandas scikit-learn feature\_engine matplotlib seaborn

### 3. Dataset

* **Input Data**: The script reads data from a CSV file named titanic\_train.csv.
* **Target Variable**: The column survived is used as the target variable, representing whether a passenger survived (1) or not (0).
* **Features**: The primary features include age, fare, sex, embarked, and family, which is a derived feature combining the number of siblings/spouses and parents/children aboard.

### 4. Data Cleaning

The data cleaning process involves the following steps:

* **Dropping Irrelevant Columns**: Columns such as passengerid, name, ticket, and cabin are removed as they are not useful for predicting survival.
* **Feature Engineering**: A new feature family is created by summing the sibsp (number of siblings/spouses) and parch (number of parents/children) columns. The original sibsp and parch columns are then dropped.

### 5. Data Preprocessing

Different preprocessing steps are applied to the features using pipelines:

* **Age Pipeline**: Imputes missing values with the median, caps outliers using Winsorization, and scales the data using StandardScaler.
* **Fare Pipeline**: Applies Winsorization to handle outliers and scales the data using StandardScaler.
* **Embarked Pipeline**: Fills missing values with the most frequent category, encodes the categorical data using count encoding, and scales it using MinMaxScaler.
* **Sex**: Encoded using OneHotEncoder to handle categorical data.
* **Family**: Scaled using MinMaxScaler.

These preprocessing steps are integrated into a ColumnTransformer to ensure that the appropriate transformations are applied to the corresponding columns.

### 6. Model Definition

The script defines and trains the following machine learning models:

* **Logistic Regression**: A linear model for binary classification.
* **Naive Bayes**: A probabilistic classifier based on Bayes’ theorem.
* **Support Vector Machine (SVM)**: A classifier that constructs hyperplanes to separate different classes.
* **Random Forest**: An ensemble model that combines multiple decision trees.

Each model is implemented as a pipeline, with the preprocessing steps applied before the model training.

### 7. Model Evaluation

The performance of each model is evaluated on a test dataset using the following metrics:

* **Accuracy**: The proportion of correctly predicted instances.
* **Precision**: The proportion of positive predictions that are actually correct.
* **Recall**: The proportion of actual positives that are correctly identified.
* **F1 Score**: The harmonic mean of precision and recall, providing a balanced measure.

The results are stored in a dictionary and later converted into a pandas DataFrame for easy comparison.

### 8. Results

The evaluation results for each model are compiled into a DataFrame and displayed. The results include the accuracy, precision, recall, and F1 score for each model, allowing for a straightforward comparison of model performance.

### 9. Visualization

A bar plot is generated to visualize the performance of the models across the evaluation metrics. The plot provides a clear comparison of how each model performed relative to the others. The plot is saved as model\_comparison.png in the current working directory.

### 10. Execution

To execute the script, ensure that the Titanic dataset (titanic\_train.csv) is available in the working directory. Running the script will train the models, evaluate their performance, and generate a comparison plot. The results will be printed in the console, and the plot will be saved as a PNG file.

### 11. File Output

* **model\_comparison.png**: This file contains a bar plot comparing the accuracy, precision, recall, and F1 score of the Logistic Regression, Naive Bayes, SVM, and Random Forest models. The plot is generated using matplotlib and seaborn, providing a visual representation of the model performance.