Titanic data

**Problem statement:**

The dataset includes the following columns:

#description of columns survived = survived status in the titanic ship

**pclass** = passenger class in the titanic ship

**sex** = mal / female

**age**

**sibsp** = individual is traveling with sibling or spouse

**parch** = individual is traveling with parent or children

**fare** = ticket fare of titanic ship

**embarked** = onbording location of the titanic ship

**class** = passenger class of the titanic ship / repeated column as pclass

**who** = repeated column as sex

**adulte\_male** = repeated column as sex

**deck** = deck of the ship. we will have to check whether the deck position is related to survival status or not.

**embark\_town** = repeated column as embarked

**alive** = repeated column as survived

**alone** = repeated column as sibsp and parch

The aim of the project is to predict if there are survivors on the titanic ship (yes/no). This is a classification model, because the target variable is categorical

**Data Analysis Journey**

Step 1 : after loading the data we do the initial inspection to understand the structure of the dataset

Step 2 : then we are checking the count of missing values

Step 3 :#handling categorical data after segregating the categorical and continuous columns

We are listing out the categorical columns and applying encoding to the relevant string columns

Step 4 : feature engineering

We are modifying and creating new features as necessary. We are also combining distinct categories of similar type into a parent category to make the data concise

Step 5 : feature scaling we are standardizing the numerical features such that the model can capture the patterns accurately

Step 6 : dividing into train test split to split the data for building the model

**Plot :**

using seaborn's countplot() function to create a countplot for categorical variables

using seaborn's boxtplot() function to create a boxtplot for continuous variables

**Splitting the Dataset**

• Split the dataset into training and test sets to make predictions using the test data breaking dataset to train and test to 75% train data and 25% test data

Applying Machine Learning Models

• Applied Logistic Regression model, decision tree

* Comparing Model Accuracy and Tuning

• Cross-Validation Without Hyperparameter Tuning

• A DecisionTreeClassifier is initialized, and cross-validation is performed using StratifiedKFold to ensure balanced splits of the dataset.

• Cross-validation scores are printed along with their mean

• Cross-Validation with Hyperparameter Tuning:

• A parameter grid is defined for hyperparameter tuning.

• GridSearchCV is used to perform an exhaustive search over specified parameter values for the DecisionTreeClassifier using cross-validation.

• The best parameters and cross-validation score are printed.

• Cross-validation scores are printed for the best model along with their mean.

• Visualization:

• The decision tree is visualized for the best model obtained from hyperparameter tuning using plot\_tree.

* Evaluate the best model on the test set
* Confusion Matrix with interpretation , precision and recall
* Applied random forest classifier
* Cross-validation scores are printed along with their mean
* Cross validation with hyperparameter
* Feature importances with the help of random forest
* Training the random forest classifier model
* Getting feature importances
* Creating a data frame for feature importances
* Ploting the feature importances

**Conclusion**

• While applying decision tree we got overfitting scenario, so we refitted the decision tree model with customization of parameters

• Considering the accuracy scores of Logistic Regression (with cross-validation and hyperparameter tuning along with their default scores), we concluded that the model gave the best accuracy score: 0.81

Best Model Metrics: decision tree

Accuracy: 0.81

Precision: 0.812

Recall: 0.79

F1 Score: 0.84