Titanic - Final Project Report

Ella Liang
Github: https://github.com/ellayuanjian/proj-titanic

Introduction

In this project we want to answer the question: “what categories of people were more likely to survive in the Titanic shipwrecks?” using the dataset which comes from Kaggle and was collected from the internet that contains passenger data. This question is intriguing because some groups of people seem more likely to survive than others besides the element of luck.

There are 12 columns in the dataset. The target variable is survival status (“Survived” variable), “Survived” description: Categorical, 0 = No survival, 1 = survival. Since a passenger could only survive or not survive, this problem is a classification problem. There are two id columns: PassengerId: Numerical, a unique id for each passenger, do not have a unit, and Name: Name of the passenger. There are 9 features and 1309 data points. However, we mainly focus on the 891 data points in the train.csv.

Features description: Pclass：Categorical, 1 = 1st class, 2 = 2nd class, 3 = 3rd class. Sex: Categorical, female or male. Age: Numerical, age of the passenger, unit is a year. SibSp: Numerical, number of siblings or spouses aboard the Titanic, unit is a person. Parch: Numerical, number of parents or children aboard the Titanic, unit is a person. Ticket: ticket number, do not have a unit. Fare: Numerical , passenger fare. Cabin: cabin number, do not have a unit. Embarked: Categorical, C = Cherbourg, Q = Queenstown, S = Southampton.

In *A Statistical Analysis & ML workflow of Titanic,* author Masum Rumi worked us through the basic supervised machine learning process of the Titanic dataset. He tried to predict the survival status of passengers in the test set. His main goal is to predict the survival status of each passenger in the test set. He found female and upper class passengers have higher survival rates. He got a test score: 0.81339 out of 1.

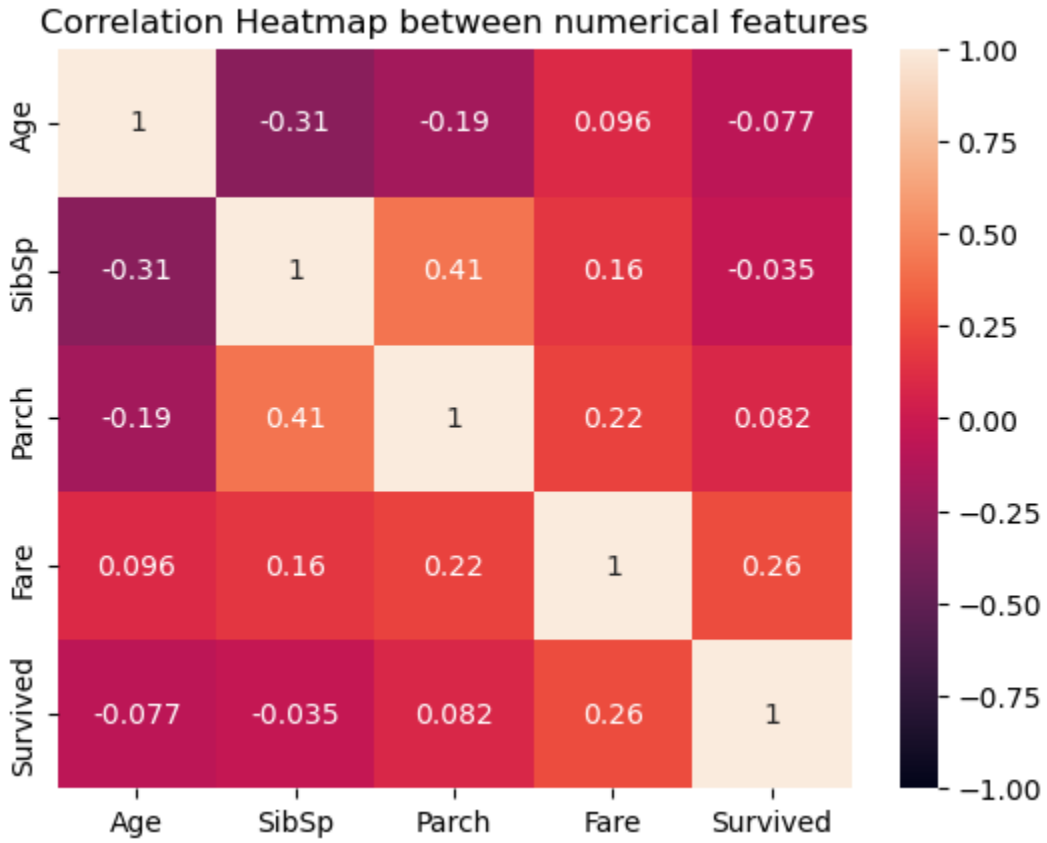
In *Titanic Data Science Solution,* author Manav Sehgal tries to answer the question: “can their model determine based on a given test dataset not containing the survival information, if these passengers in the test dataset survived or not.” He found it is helpful to create a new feature “Family Size” from “Parch” and “Sibsp”, and then drop these two features. He also found higher fare paying passengers, female, and infants had higher survival rate. His test score is 86.76 out of 100.

EDA

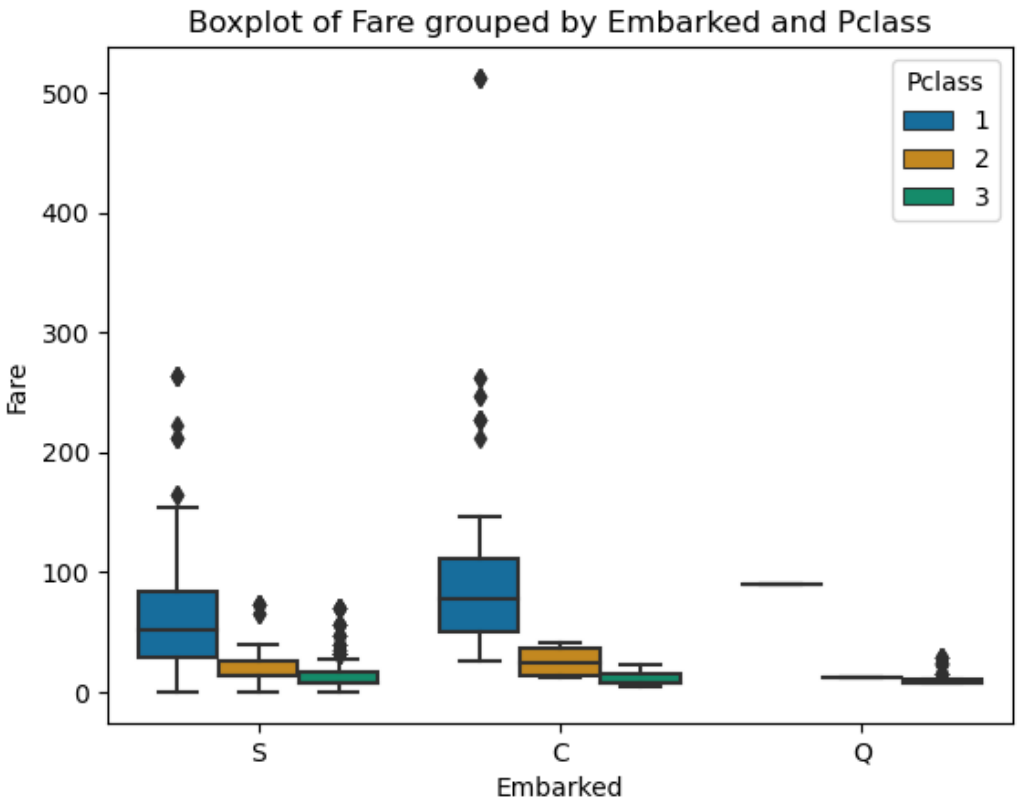
I found 5 important figures during my EDA process, these 5 figures helped me understand the Titanic dataset better.

Below histogram shows distribution of each numeric feature including the target variables. The visualization of the target variable is the first histogram. The balance of

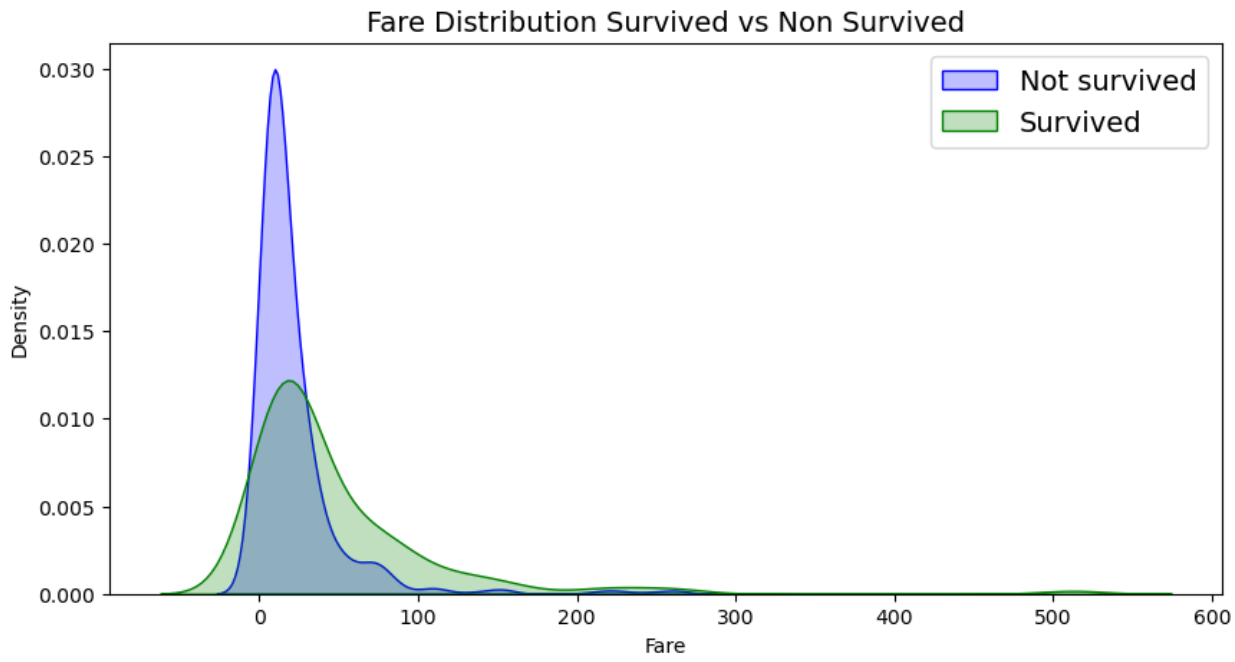
the survived class (Survived = 1) is around 38.38%, and the balance of the non survived class (Survived = 0) is around 61.62%.



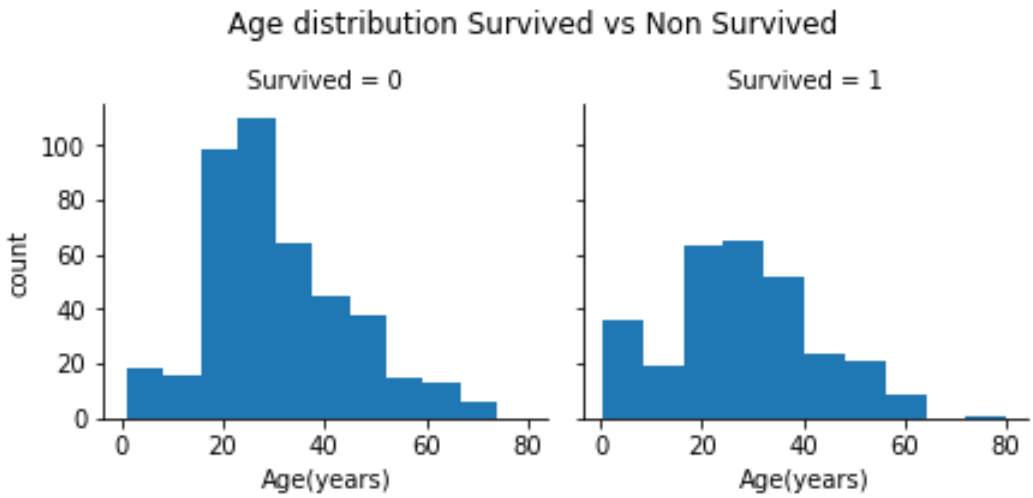
From this graph we can see there is no perfect correlation or too high correlation between numerical features, so we don’t need to drop any numerical features.



From this boxplot, we can see that most of the passengers who embarked at “C” paid higher fare than people who Embarked at “S” and “Q”, most of the passengers who embarked at “Q” paid the lowest fare among these 3 locations. Moreover, we can see 1st class passengers paid the highest fare among 3 Pclass, and 3rd class passengers paid the lowest fare. Most of the passengers embarked at location C.



From this graph, we can see that most of the survived passengers paid higher fare than non survived passengers.



From this graph, we can see that a large number of 15-25 year olds passengers did not survive, and most passengers are in the 15-35 age range.

Methods

- Splitting

The Titanic dataset is IID, not a time-series data, and does not have group structure, since each entry of the dataset recorded the information for a unique individual, and the information does not change over time.

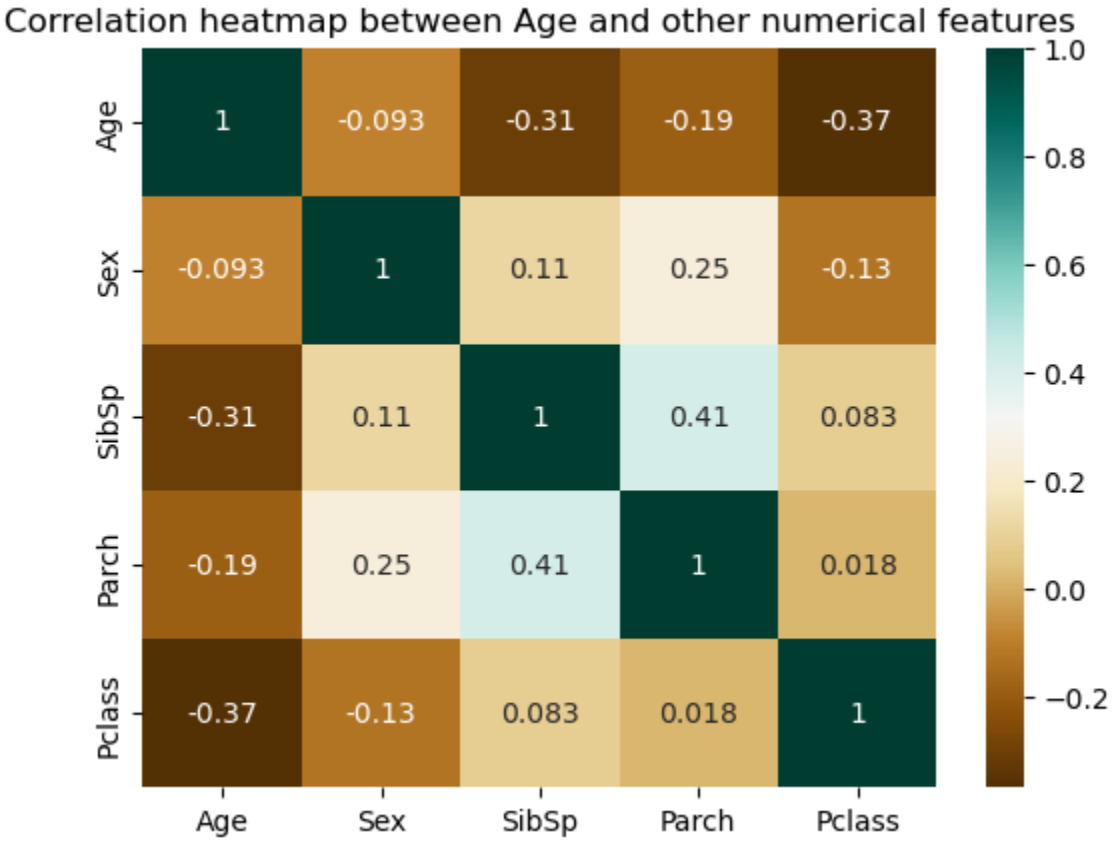
I choose the basic approach for splitting the data, since the dataset fulfilled the i.i.d assumption. I also use the KFold cross validation here, because when we use grid

search, KFold cross validation is useful for tuning hyperparameters. The dataset is not imbalanced, stratified splits are not needed here.
Because “Ticket” has too many non-numeric unique elements and is not very useful for the prediction, I choose to drop this column to ease the analysis.

- Data Preprocessing

We have missing values in “Age”, “Cabin”, and “Embarked”. Onehot encoder will help us deal with “Embarked” feature.

“Age” seems essential in predicting survival status; simply replacing missing values with mean, median, or mode will hugely impact our prediction. We change “Sex” to binary values, 0 means male and 1 means female. By plotting the correlation heatmap between Age and other numerical features, we can get “Age” is negatively correlated with “Pclass”, “Parch” and “SibSp”. Thus, we replace missing values in “Age” features by using these 3 features. We replace a missing value by the median of its similar rows, if it doesn’t have similar rows, we replace it with the median of the “Age” feature.



In the “Cabin” column, I notice that each non-null entry always starts with a single letter and is followed by numbers, so I replace each entry by the starting letter. There is a connection between “Fare” and “Cabin”, so by calculating the mean of the “Fare” grouped by “Cabin”, we can replace the null values in “Cabin” by appropriate cabin initials.

I chose OneHotEncoder for “Cabin”, and “Embarked” because they are categorical features. I choose StandardScaler for “Age”, “SibSp”, “Parch”, and “Fare” because they are continuous features. Last but not least, I choose OrdinalEncoder for “Pclass” because it is ordinal feature and it has ordered categories. After preprocessing data, we have 17 features.

- ML pipeline

I choose 4 different ML algorithms. First, I implemented the simple algorithm, Logistic Regression. I tune the “C”, “solver” and “penalty” hyperparameters. I tried 100, 10, 1.0, 0.1, and 0.01 for “C”, which is the inverse of regularization strength. I tried “none”, “l1””, “l2”, and “elasticnet” for “penalty”.Lastly, I tried solver = ['newton-cg', 'lbfgs', 'sag','saga'] for penalty = ‘none’, solver = ['newton-cg', 'lbfgs', 'sag','saga','liblinear'] for penalty = ‘l2’, solver = ['saga'] for penalty = ‘elasticnet’, and solver = ['liblinear'] for penalty = ‘l1’. Secondly, I implemented a more complex algorithm, K nearest neighbors (KNN) classification. I tuned the “n_neighbors” hyperparameter. I tried values 1, 3, 5, 10, and 20. Thirdly, I implemented support vector machine classification. I tuned the “gamma” and “C” hyperparameters. I tried 0.001,0.01,0.1, and 1 for “gamma”, and 1,10,50,100,300, and 1000 for “C”. Last but not least, I implemented Random Forest Classification. I tuned “n_estimators”, “max_features”, “min_sample_split”, and “min_samples_leaf” hyperparameters.

I used the roc_auc (the area under the ROC curve) metric to evaluate models’ performance because I equally care about survived and non-survived class, and my model is not very balanced (but it is not imbalanced), auc_roc metric is better than accuracy metric.

In my ML pipeline, I loop through 10 different random states and get the best model and best test score for each random state. After getting these best models and test scores, I calculate the mean and standard deviation of test scores in order to measure the uncertainties due to splitting and non-deterministic ML methods.

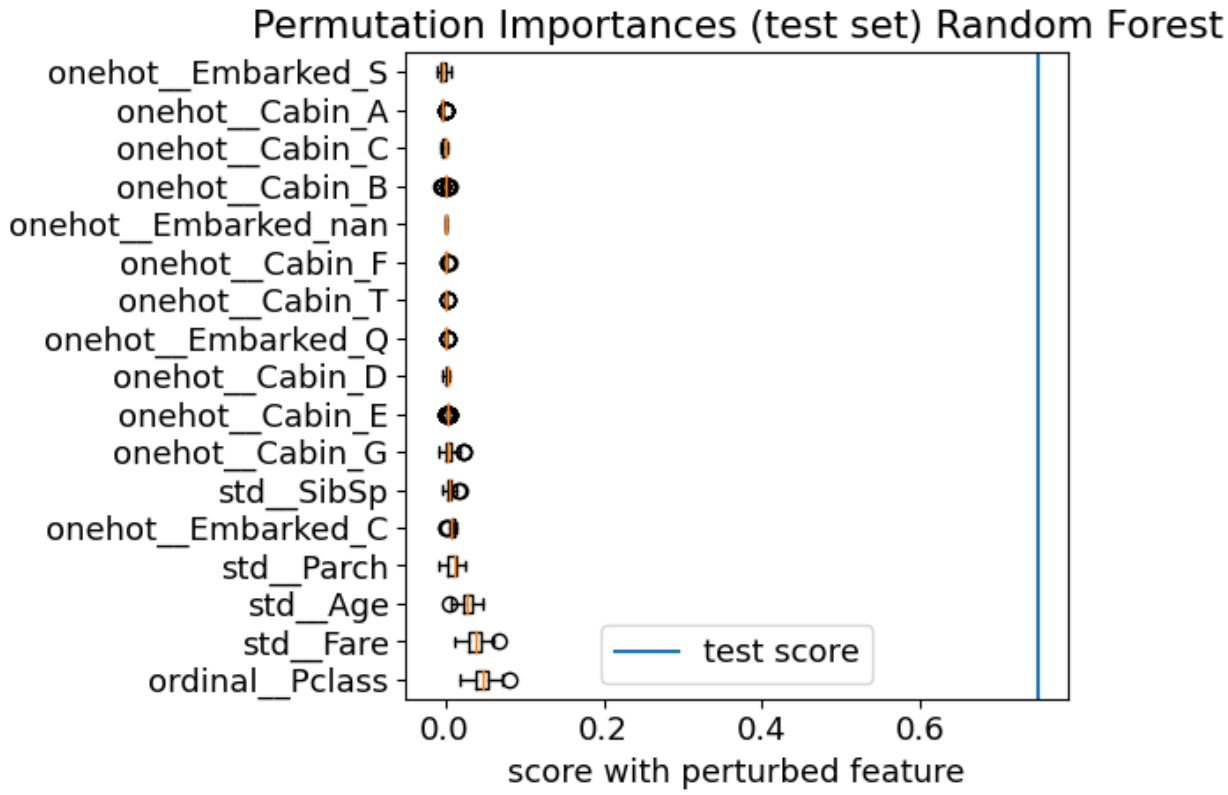
Results

The mean baseline roc_auc score is 0.5, which is worse than all four algorithms models. The mean of baseline accuracy is 0.61. Mean of baseline f_beta score when beta = 0.5 is 1.89; mean of baseline f_beta score when beta = 1 is 0.75; mean of baseline f_beta score when beta = 2 is 0.47.

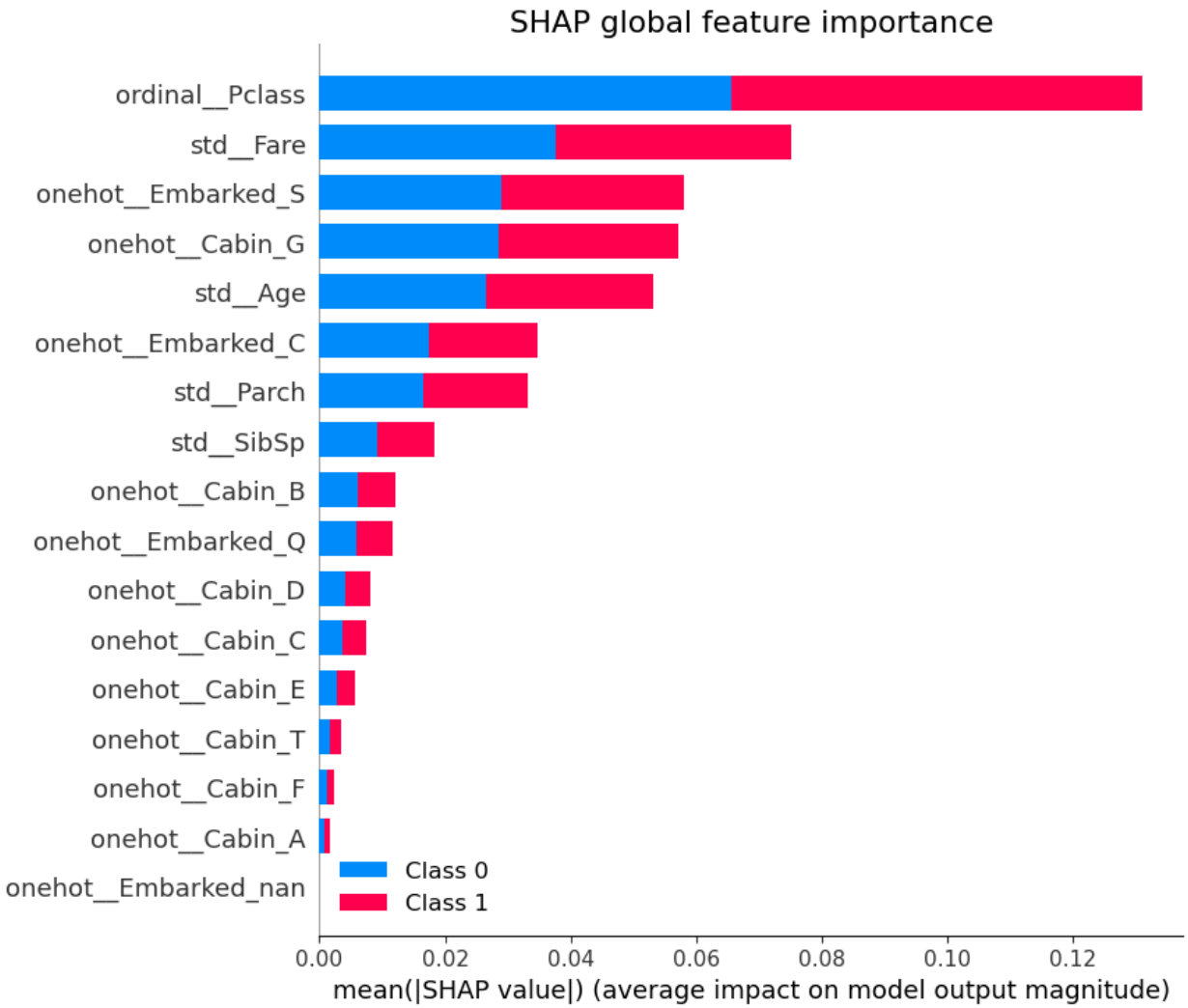
	Algorithm	Test score mean	Test score std
3	Random Forest Classifier	0.721289	0.020783
0	Logistic Regression	0.717927	0.022270
1	KNN Classifier	0.713445	0.022620
2	SVM Classifier	0.711204	0.017333

From the summary performance table above, we can see that the Random Forest classifier model is most predictive.

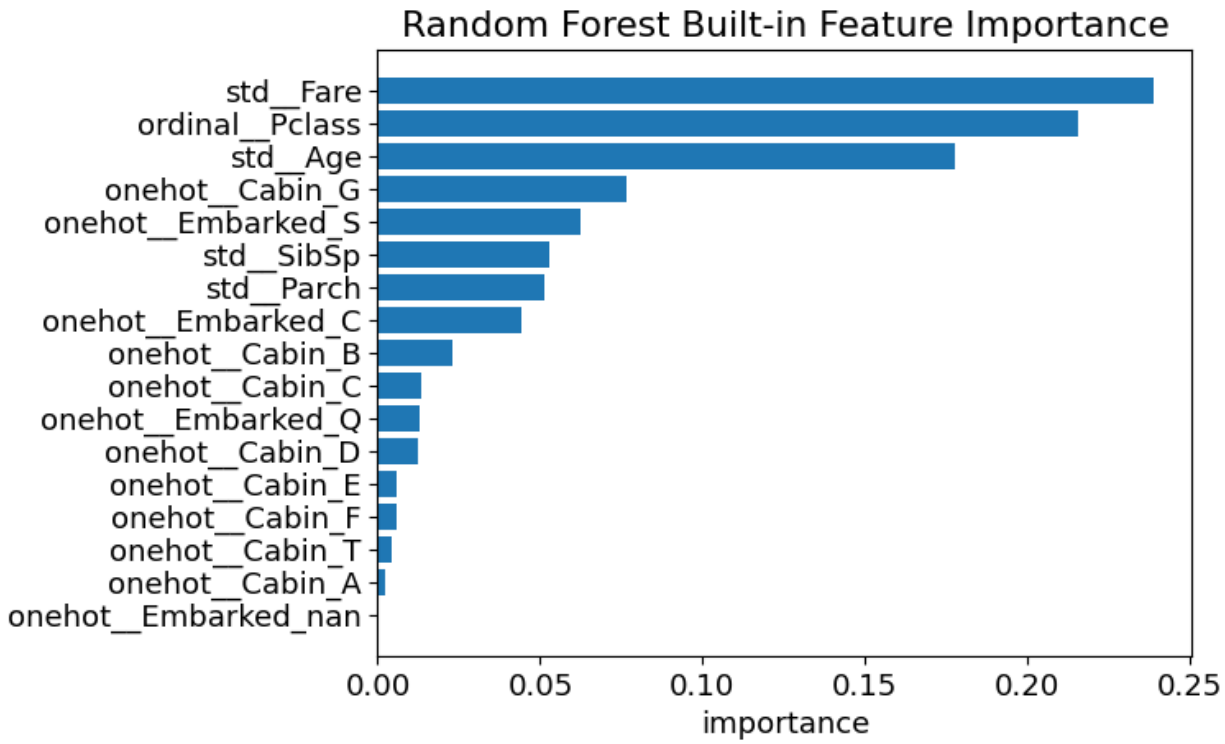
I calculate three global feature importances for my best model,: permutation importance, SHAP global importance, and the Random Forest built-in feature importance.



From permutation importances, we get ordianl_Pclass, std_SibSp, and std_Age features are the top 3 global important features.



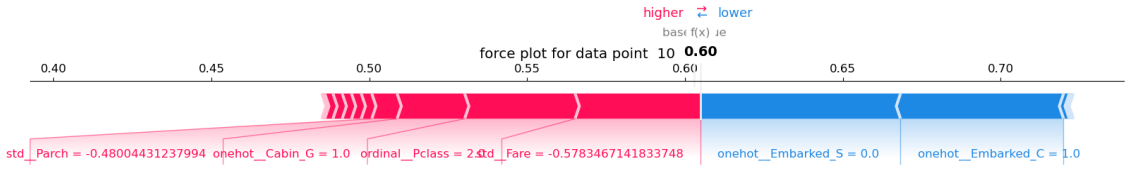
From the SHAP global method, we can get ordinal_Pclass, std_Fare, and onehot_Embarked_S features are the three most important features.



By using the built-in feature_importances_ function for Random Forest classification, we get std_Fare, ordinal_Pclass, and std_Age features are the three most important features.

After combining these three result graphs, we can get “ordinal_Pclass” is the most important feature because it is listed in top 3 important features for all three ways, “std_Age” is the next most important feature because it is listed in top 5 important features for all three ways . On the other hand, “onehot_Cabin_F” is the least important feature because it is listed in the top 3 least important features for all three ways. “onehot_Embarked_nan” feature does not have any global importance in the last two methods and has little importance using the first method, this is because “one_Emarked_nan” is only important for the two missing values in the “Embarked” feature. Almost all “onehot_Cabin” features only have little importances except the “onehot_Cabin_G”.

I also calculate several data points’ SHAP values for local feature importance. By drawing force plots for these data points, we can see which features have negative and positive contributions to these instance’s predictions.



The force plot above is one of the force plots I draw. In this force plot for index = 10, we can see that “onehot_Embarked_S” has the largest negative contribution and “std_Fare” has the largest positive contribution to the prediction among all features.

Outlook

In this project, I only implement four algorithms. There are other classification algorithms that may create better models than these four algorithms. If I have more time, I would like to try the gradient boosting classification algorithm and decision tree algorithm. The weak spots of my modeling approach is I choose only a little amount of values for hyperparameter tuning. Because trying too many values for hyperparameters will take a lot of time to execute, I only choose several values that I think are reasonable and potentially good values that will produce high test scores. Also, combining features might generate other useful features for modeling, for example, creating a new feature “Alone” which indicates whether a passenger is alone by using “Parch” and “Sibsp” features. I can also drop some features that are not important after calculating global feature importances. We can also check whether outliers exist in the dataset because outliers will greatly affect our prediction. If we can collect the location of each cabin or the distance between each cabin and the exit, we can improve the model performance because generally, if a cabin is close to the exit, passengers inside the cabin have higher survival rate. In order to improve the interpretability, we can use LIME or Local Interpretable Model-Agnostic Explanations to explain predictions.

Reference

A Statistical Analysis & ML workflow of Titanic - Masum Rumi,
https://www.kaggle.com/code/masumrumi/a-statistical-analysis-ml-workflow-of-titanic

Titanic Data Science Solution - Manav Sehgal,
https://www.kaggle.com/code/startupsci/titanic-data-science-solutions

Titanic - Machine Learning From Disaster
https://www.kaggle.com/c/titanic