Case Study of Titanic:

This case study showcases the development of a binary model to predict the probability of survival in the loss of Titanic. I demonstrate the overall modeling process, including preprocessing, exploratory analysis, model fitting, adjustment, and interpretation as well as other relevant techniques such as imputation for missing data.

Introduction

The sinking of RMS Titanic has brought to numerous machine learning competitions a quintessential dataset. The unsinkable British passenger liner struck an iceberg on 15 April 1912 in her maiden voyage, and was eventually wrecked. More than 1500 people perished in the great loss. Decades of effort has been devoted to the study the tragic accident, in which one major interest for statistical inquiries is to model and predict the probability of survival given personal characteristics. In recent years the web has witnessed the birth of numerous variants of Titanic data, with one primary source being Encyclopedia Titanica (1999), a site started in 1996 as an attempt to tell the story of every person that traveled the Titanic as a passenger or crew member. This case study grows from the most up-to-date version of the site's data as of October 2020, with the following columns available.

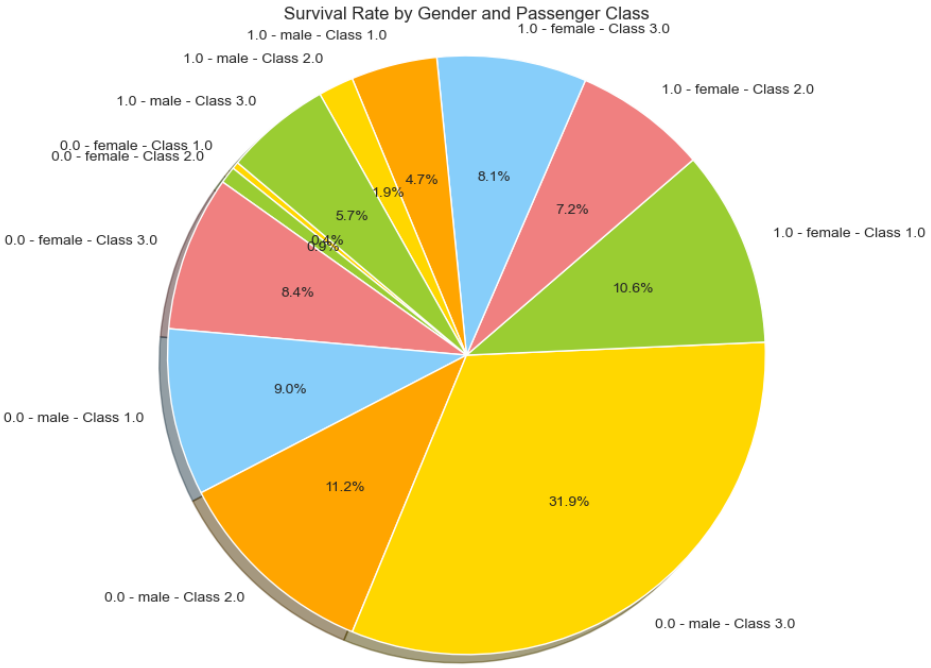
Variable	Definition	Note
Survived	Survival Status	0 = No, 1 = Yes
Name	Passenger Name	
Pclass	Passenger Class	1 = 1st, 2 = 2nd , 3 = 3rd
Sex	Gender	
Age	Age in years	
Sibsp	# of siblings / spouses aboard the titanic	
Parch	# of parents / children aboard the titanic	
Fare	Passenger Fare	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Exploratory Data Analysis:

In Exploratory Data Analysis (EDA), I begin by understanding the data's shape, basic information, and descriptive statistics. This helps me get an initial sense of the data. Next, I check for null values in the dataset. If any features have missing values, I perform data imputation to clean the data, a step which I'll discuss further in the data cleaning section.

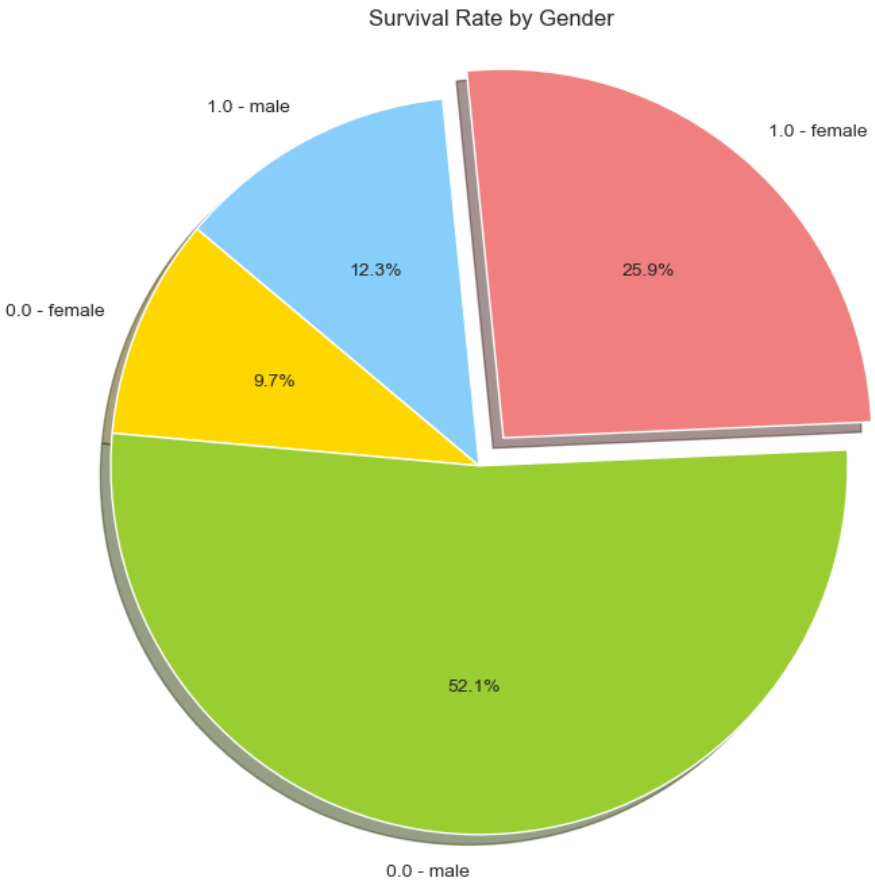
However, these initial steps only provide surface-level insights. To gain a deeper understanding, I move on to univariate and bivariate analysis. During this analysis, questions naturally arise, such as:

a. What factors influenced the likelihood of survival?



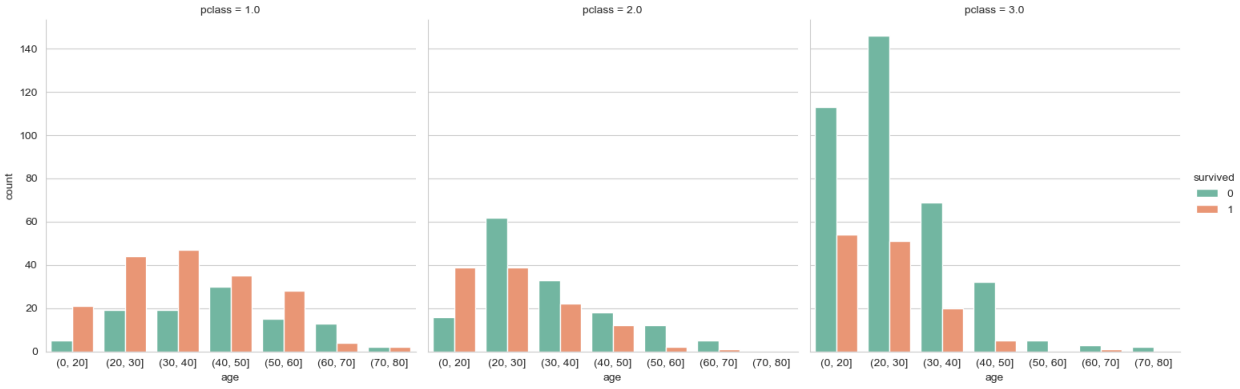
Based on my analysis, it appears that females in the first class had the highest chance of survival during that event. Specifically, I found that 10.6% of females in the first class survived. This suggests that being a female in the first class greatly increased the likelihood of survival on that night.

b. Did gender play a role in survival rates?



Yes, gender did seem to play a role in the survival rate. The pie chart indicates that 25.9% of females survived, suggesting that a larger proportion of females survived compared to males. This observation supports the idea that gender had an impact on survival rates during that event.

c. Did age or passenger class affect survival rates?



I don't believe age or passenger class had a significant effect on the survival rate. Looking at the bar chart, we can see that although there were more people in the third class compared to

other classes, the difference in survival rates wasn't substantial. This suggests that factors like age or passenger class didn't strongly influence who survived and who didn't during that event.

By answering these questions through thorough analysis, I gain deeper insights into the data. This structured approach allows me to uncover patterns and relationships within the dataset, leading to a more comprehensive understanding of the underlying factors influencing survival rates.

Data Cleaning:

During the process of handling missing values, most features in the dataset didn't have any null values except for the 'age' feature, which had around 20% missing values. Dropping these missing values wasn't an option, so we decided to use data imputation.

Initially, we used the median method to fill the NaN values in the 'age' feature. However, this approach created outliers in the data. To avoid this issue, we researched alternative methods and found that using other features to fill the NaN values could be effective.

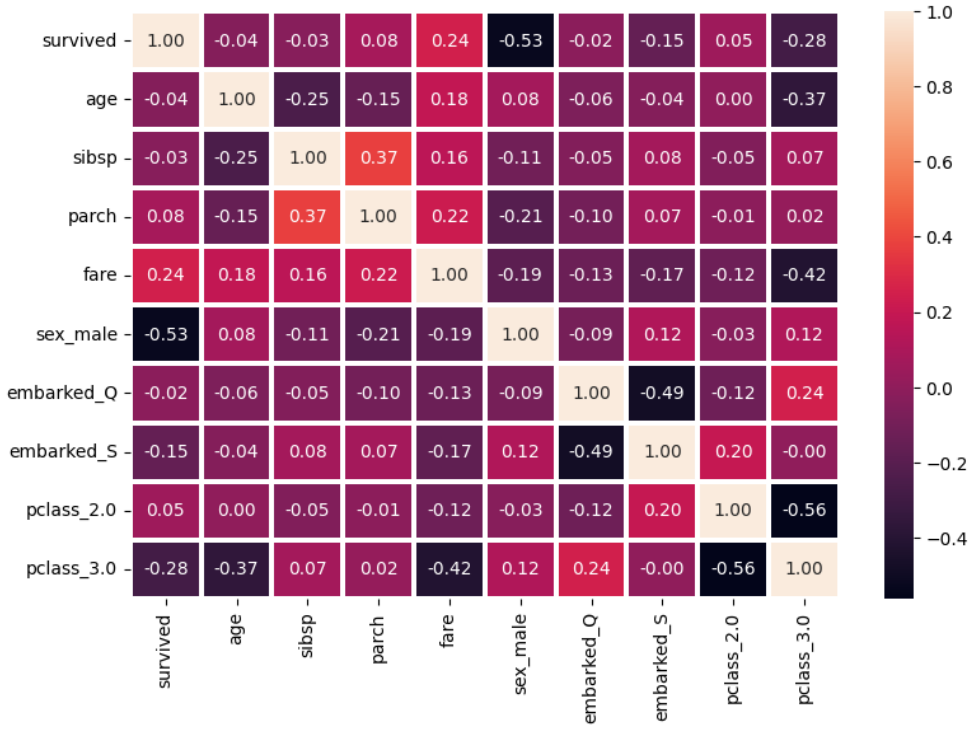
We started by extracting the title of each person from their name, such as 'Miss', 'Mrs', 'Mr', 'Master', and various titles indicating royalty or officer status. Since there were many different titles, we simplified them into basic categories.

Afterwards, we analyzed the ages associated with these titles and found that this approach yielded promising results. We then calculated the mean age for each title, taking into account gender and passenger class.

By using this method for data imputation, we were able to fill in the missing age values more accurately and effectively, ultimately improving the quality of our dataset.

Model Building

Before I begin training a model, I first organize the data. This means I change things like names or categories into numbers because computers like numbers more than words. Once everything's in numbers, I then look to see if some numbers change together. For example, if one number goes up, does another one also go up? This is called checking for correlation. So, in short, I start by turning words into numbers, and then I see if certain numbers are linked to each other.



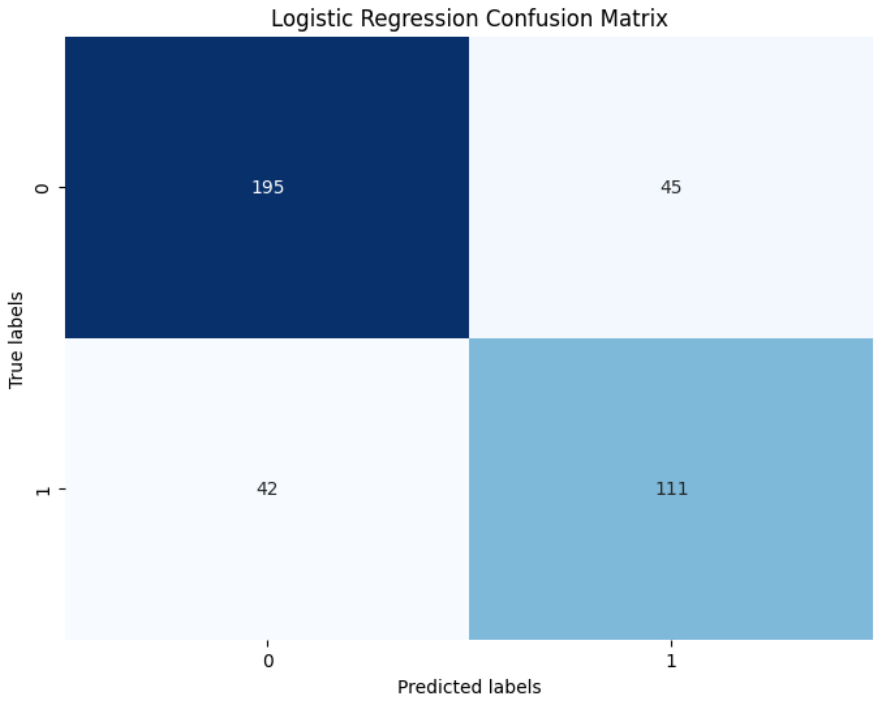
After that, I utilized the standard scaler from sklearn to standardize the data between -1 and 1. This was done to provide clearer data insights and enhance the model's performance.

Then I test the following classifiers from scikit-learn:

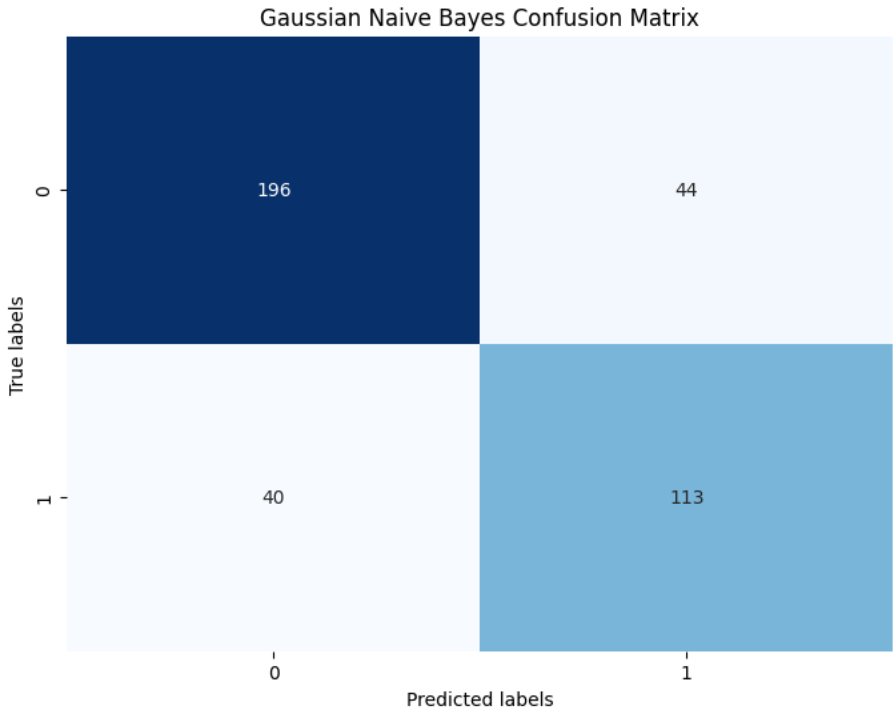
- Logistic Regression
- Gaussian Naive Bayes
- K Nearest Neighbors KNN
- Decision Tree Classifier
- Random Forest Classifier
- SVM Classifier

For comparison of the result, I use these metrics accuracy_score, classification_report, confusion_matrix.

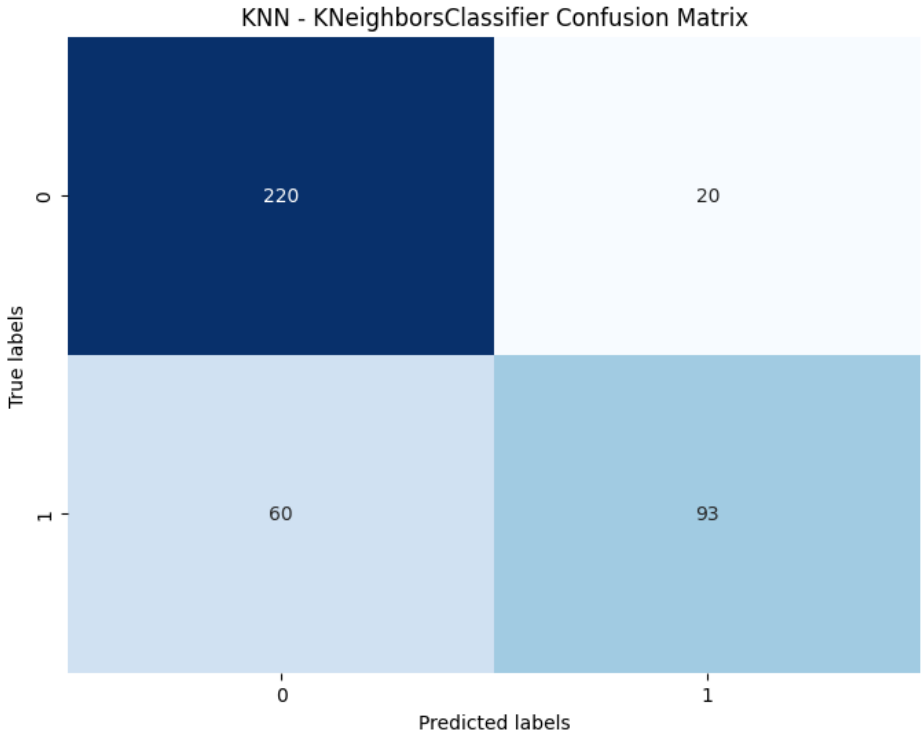
Logistic Regression:



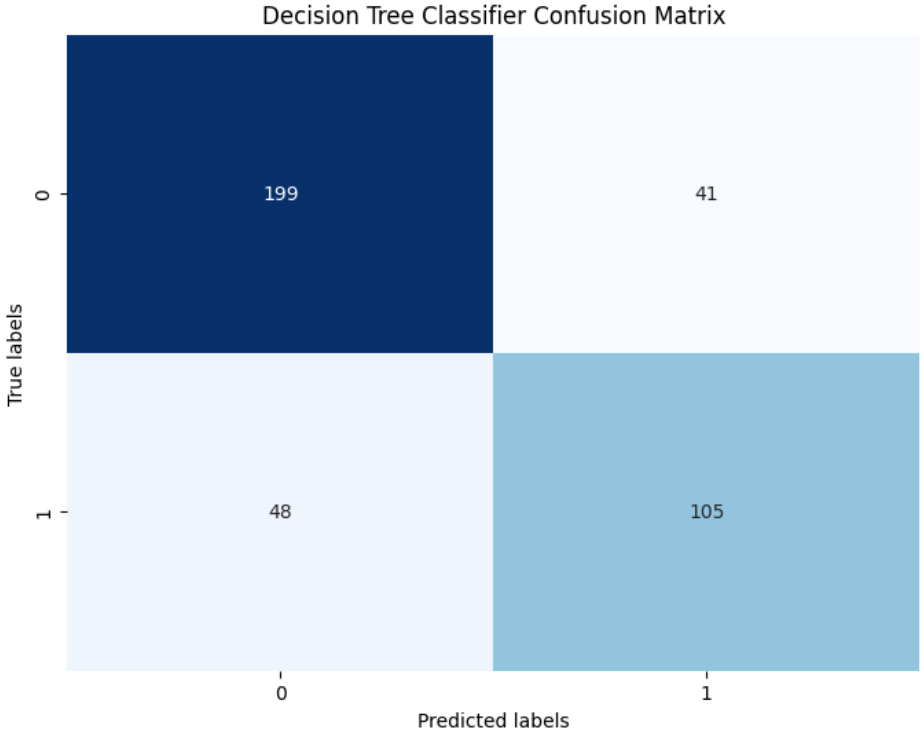
Gaussian Naïve Bayes



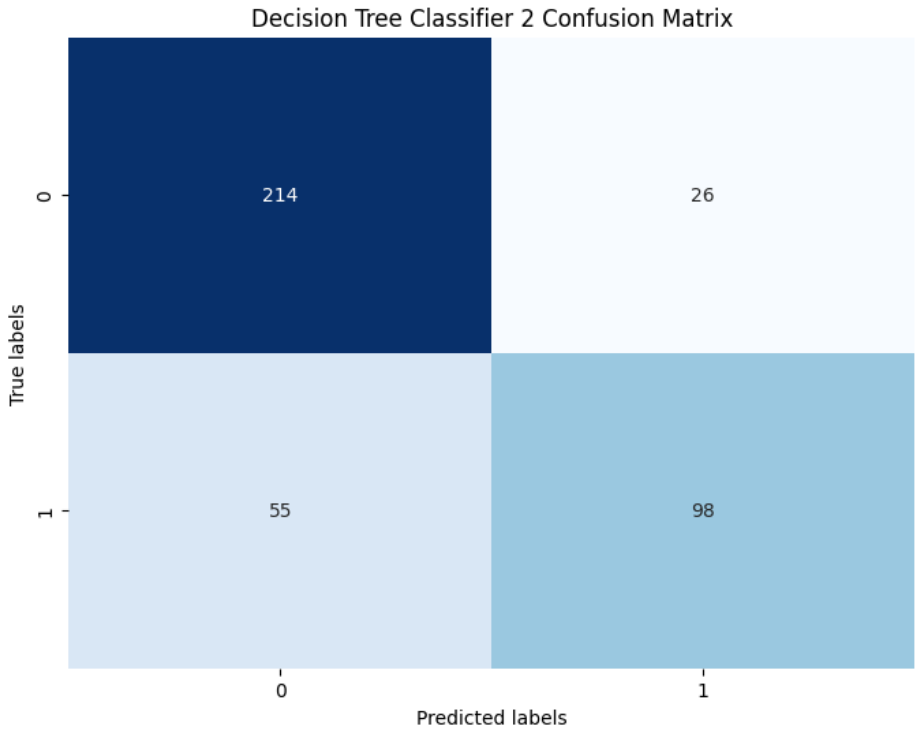
KNN – KNeighbors Classifier



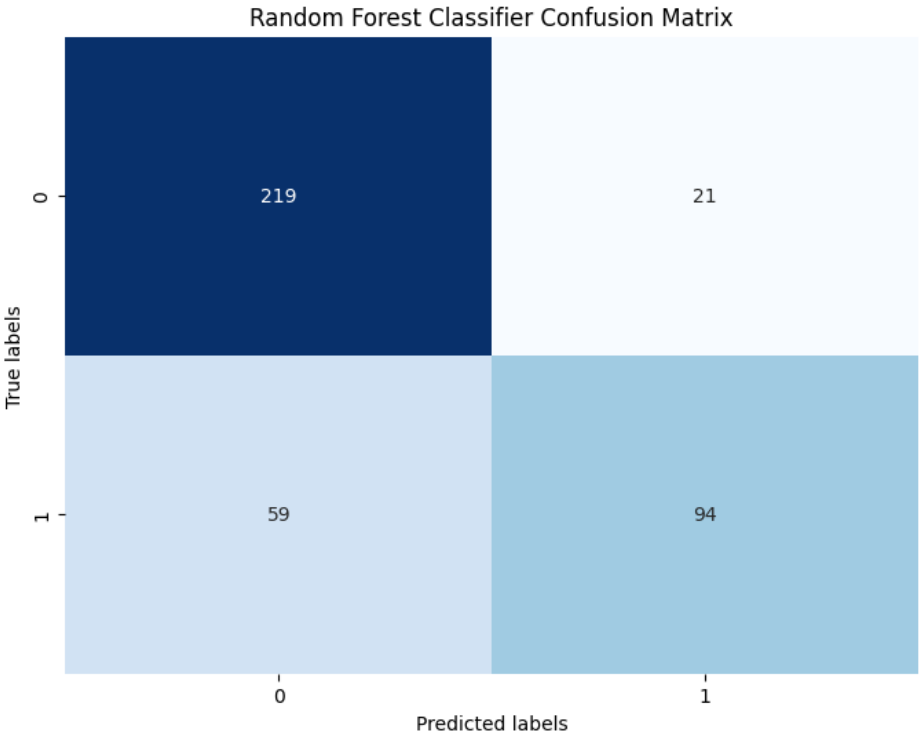
Decision Tree Classifier



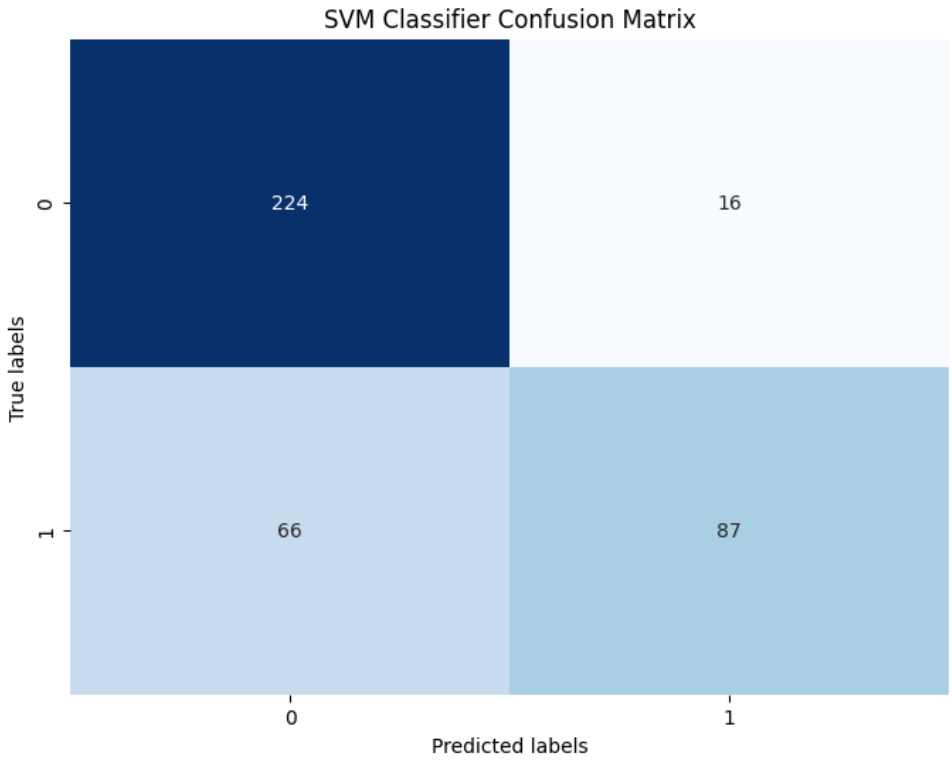
Decision Tree Classifier 2



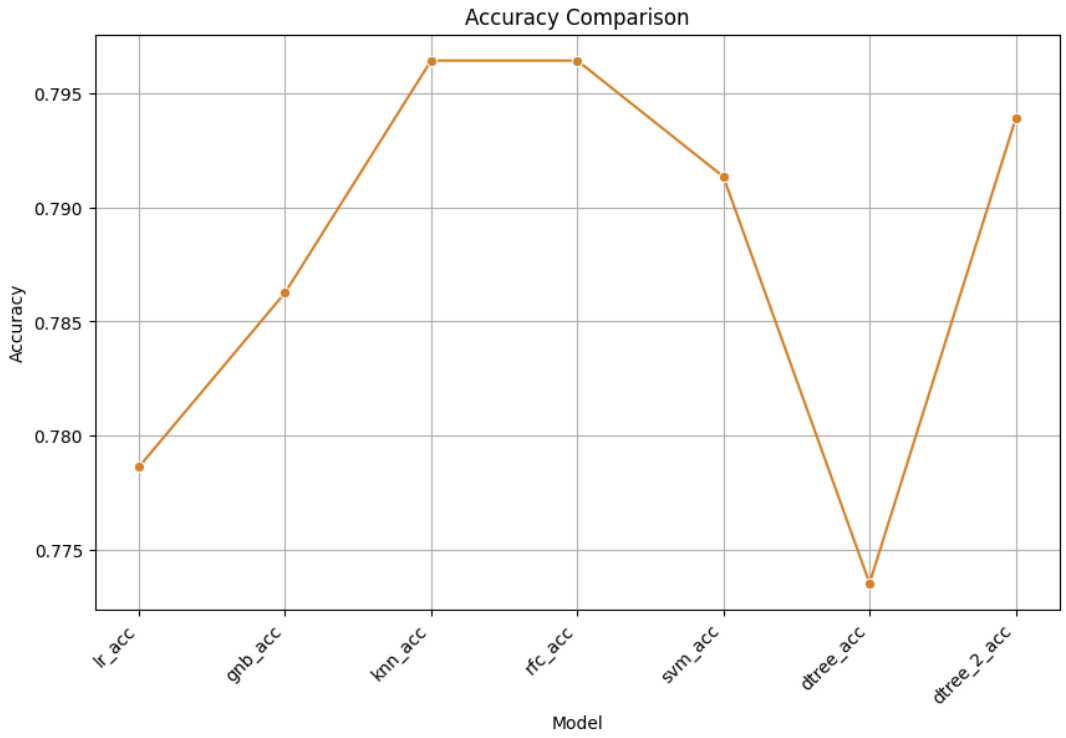
Random Forest Classifier



SVM Classifier



Accuracy Comparison of all Models



Accuracy Comparison using K Fold