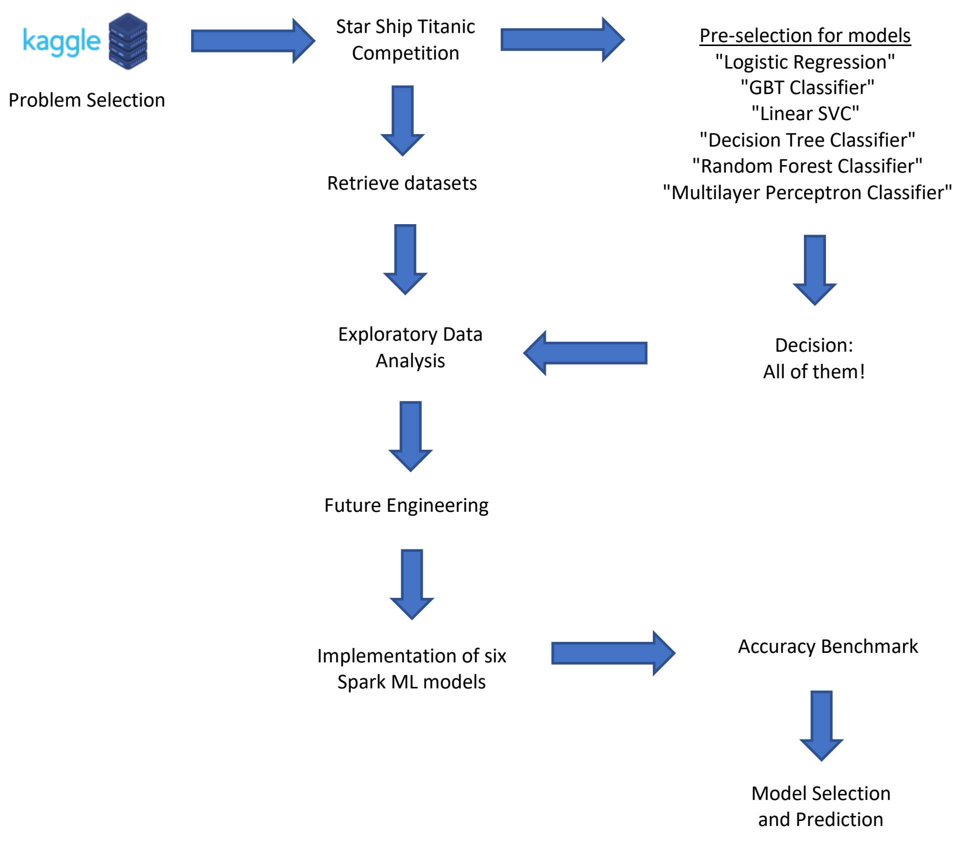
Space Ship Titanic

Architectural Decisions Document

The subject of my project is a competition on KAGGLE that will start in 2022 and continue indefinitely. A competition inspired by the “Titanic” example used for machine learning. I think it is a very effective project for comparing the effectiveness of multiple machine learning models.

In the competition, submissions are evaluated based on their classification accuracy and the percentage of correct tags predicted. Therefore, I believe that it is a project that will guide my stakeholders in both feature engineering and model selection.

1 Architectural Components Overview



Problem Selection
Star Ship Titanic Competition
Pre-selection for models
"Logistic Regression"
"GBT Classifier"
"Linear SVC"
"Decision Tree Classifier"
"Random Forest Classifier"
"Multilayer Perceptron Classifier"
Retrieve datasets
Exploratory Data Analysis
Decision: All of them!
Future Engineering
Implementation of six Spark ML models
Accuracy Benchmark
Model Selection and Prediction

2 Problem Selection

I got the data I wanted to implement from kaggle.com. These are data that can be accessed for free.

Since my goal was to try different models, I made sure that my work was a data set that could be used for training. After all, since it was a competition on Kaggle, I was aware that I might encounter all kinds of difficulties.

My project is based on "Spaceship Titanic", a Kaggle Competition. It is a project about choosing the appropriate learning method after comparing the accuracy of different models with each other. Six different models such as "Logistic Regression", "GBT Classifier", "Linear SVC", "Decision Tree Classifier", "Random Forest Classifier", "Multilayer Perceptron Classifier" were applied and the one with the highest accuracy was sought. Spark ML was used in the study.

The scenario is as follows:

"Welcome to the year 2912, where your data science skills are needed to solve a cosmic mystery. We've received a transmission from four lightyears away and things aren't looking good.

The Spaceship Titanic was an interstellar passenger liner launched a month ago. With almost 13,000 passengers on board, the vessel set out on its maiden voyage transporting emigrants from our solar system to three newly habitable exoplanets orbiting nearby stars.

While rounding Alpha Centauri en route to its first destination—the torrid 55 Cancri E—the unwary Spaceship Titanic collided with a spacetime anomaly hidden within a dust cloud. Sadly, it met a similar fate as its namesake from 1000 years before. Though the ship stayed intact, almost half of the passengers were transported to an alternate dimension!

To help rescue crews and retrieve the lost passengers, you are challenged to predict which passengers were transported by the anomaly using records recovered from the spaceship's damaged computer system.

Help save them and change history!

In this competition your task is to predict whether a passenger was transported to an alternate dimension during the Spaceship Titanic's collision with the spacetime anomaly. To help you make these predictions, you're given a set of personal records recovered from the ship's damaged computer system."

At first, I was planning to choose a single classification model. However, when I examined similar "Titanic" models, it seemed more attractive to make different classification models at the same time.

I believe that my work can be stimulating for similar models.

3 Pre-selection for Models:

My project is based on "Spaceship Titanic", a Kaggle Competition. It is a project about choosing the appropriate learning method after comparing the accuracy of different models with each other. Six different models such as "Logistic Regression", "GBT Classifier", "Linear SVC", "Decision Tree Classifier", "Random Forest Classifier", "Multilayer Perceptron Classifier" were applied and the one with the highest accuracy was sought. Spark ML was used in the study.

4 Retrieve Datasets

The data set is a data set containing the information of transported persons and private individuals in the "Space Ship Titanic", taken from https://www.kaggle.com/code/gusthema/spaceship-titanic-with-tfdf/

File and Data Field Descriptions:

train.csv - Personal records for about two-thirds (~8700) of the passengers, to be used as training data.

PassengerId - A unique Id for each passenger.

HomePlanet - The planet the passenger departed from, typically their planet of permanent residence.

CryoSleep - Indicates whether the passenger elected to be put into suspended animation for the duration of the voyage. Passengers in cryosleep are confined to their cabins.

Cabin - The cabin number where the passenger is staying. Takes the form deck/num/side, where side can be either P for Port or S for Starboard.

Destination - The planet the passenger will be debarking to.

Age - The age of the passenger.

VIP - Whether the passenger has paid for special VIP service during the voyage.

RoomService, FoodCourt, ShoppingMall, Spa, VRDeck - Amount the passenger has billed at each of the Spaceship Titanic's many luxury amenities.

Name - The first and last names of the passenger.

Transported - Whether the passenger was transported to another dimension. This is the target, the column you are trying to predict.

test.csv - Personal records for the remaining one-third (~4300) of the passengers, to be used as test data. Your task is to predict the value of Transported for the passengers in this set.

5 Exploratory Data Analysis

I used a python code to explore the data and arrived at the following statistical data. (Python codes and results can be accessed from the notebook I shared.)

```
Number of Training Examples = 8693
Number of Test Examples = 4277

Training X Shape = (8693, 14)
Training y Shape = 8693

Test X Shape = (4277, 13)
Test y Shape = 4277

Index(['PassengerId',     'HomePlanet',     'CryoSleep',     'Cabin',
'Destination', 'Age',
       'VIP',  'RoomService',  'FoodCourt',  'ShoppingMall',  'Spa',
'VRDeck',
       'Name', 'Transported'],
      dtype='object')
```

```
Index(['PassengerId',    'HomePlanet',    'CryoSleep',    'Cabin',
'Destination', 'Age',
       'VIP',  'RoomService',  'FoodCourt',  'ShoppingMall',  'Spa',
'VRDeck',
       'Name'],
      dtype='object')

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8693 entries, 0 to 8692
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   PassengerId   8693 non-null   object
 1   HomePlanet    8492 non-null   object
 2   CryoSleep     8476 non-null   object
 3   Cabin         8494 non-null   object
 4   Destination   8511 non-null   object
 5   Age           8514 non-null   float64
 6   VIP           8490 non-null   object
 7   RoomService   8512 non-null   float64
 8   FoodCourt     8510 non-null   float64
 9   ShoppingMall  8485 non-null   float64
 10  Spa           8510 non-null   float64
 11  VRDeck        8505 non-null   float64
 12  Name          8493 non-null   object
 13  Transported   8693 non-null   bool
dtypes: bool(1), float64(6), object(7)
memory usage: 891.5+ KB
None
               Age   RoomService     FoodCourt  ShoppingMall           Spa  \
count  8514.000000   8512.000000   8510.000000   8485.000000   8510.000000
mean     28.827930    224.687617    458.077203    173.729169    311.138778
std      14.489021    666.717663   1611.489240    604.696458   1136.705535
min       0.000000      0.000000      0.000000      0.000000      0.000000
25%      19.000000      0.000000      0.000000      0.000000      0.000000
50%      27.000000      0.000000      0.000000      0.000000      0.000000
75%      38.000000     47.000000     76.000000     27.000000     59.000000
max      79.000000  14327.000000  29813.000000  23492.000000  22408.000000

             VRDeck
```

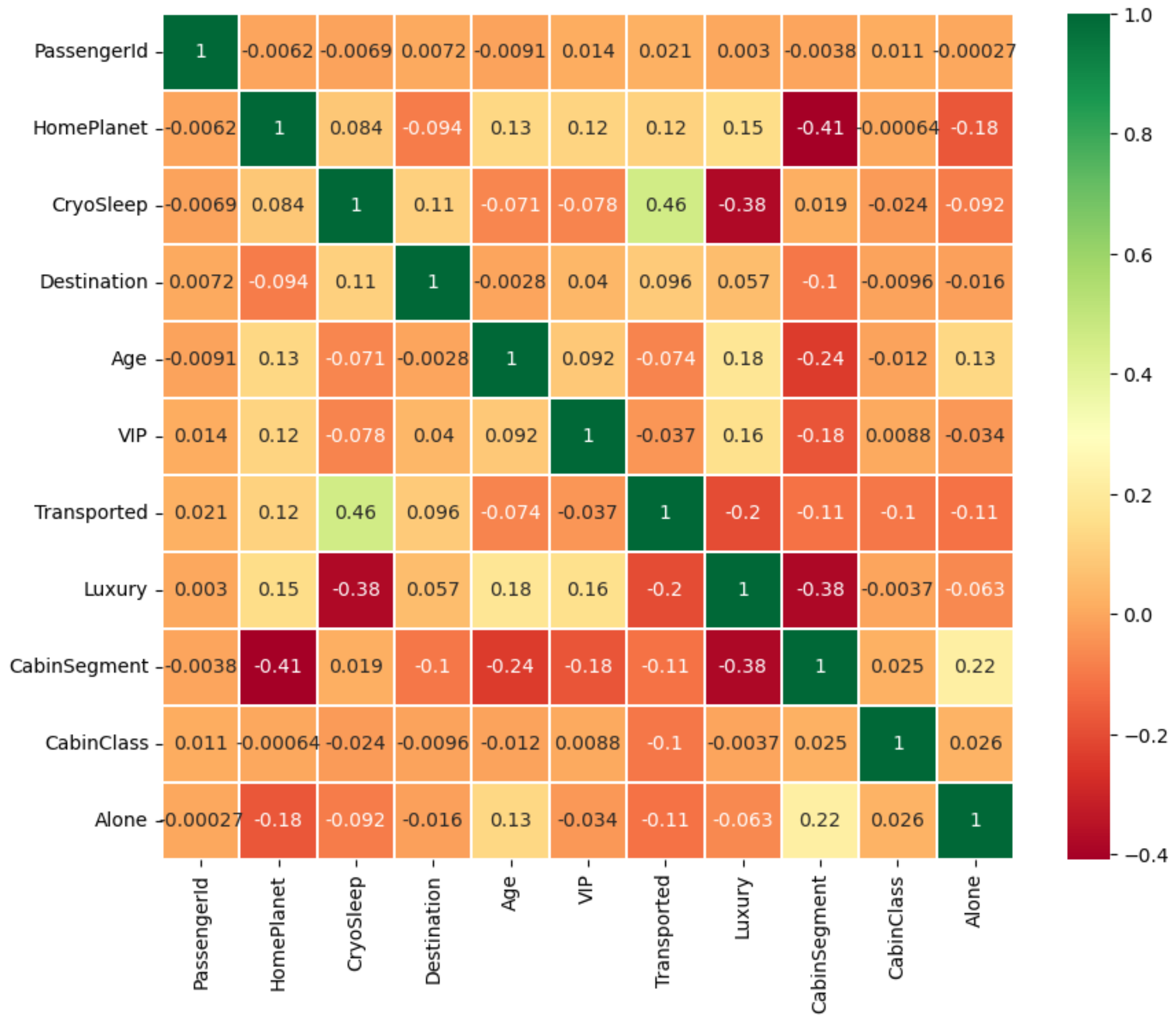
```
count    8505.000000
mean      304.854791
std      1145.717189
min         0.000000
25%         0.000000
50%         0.000000
75%        46.000000
max     24133.000000

Training Set
PassengerId column missing values: 0
HomePlanet column missing values: 201
CryoSleep column missing values: 217
Cabin column missing values: 199
Destination column missing values: 182
Age column missing values: 179
VIP column missing values: 203
RoomService column missing values: 181
FoodCourt column missing values: 183
ShoppingMall column missing values: 208
Spa column missing values: 183
VRDeck column missing values: 188
Name column missing values: 200
Transported column missing values: 0

Test Set
PassengerId column missing values: 0
HomePlanet column missing values: 87
CryoSleep column missing values: 93
Cabin column missing values: 100
Destination column missing values: 92
Age column missing values: 91
VIP column missing values: 93
RoomService column missing values: 82
FoodCourt column missing values: 106
ShoppingMall column missing values: 98
Spa column missing values: 101
VRDeck column missing values: 80
Name column missing values: 94
```

I wanted to understand which data is closely related to which data. I had to draw a heatmap graphic for this. However, such an operation required me to perform feature engineering on some data. Anyway, I could take some of this work out of the way while exploring the data, as I would need to do it in the future. (Details of the steps I took for this are in the notebook.)

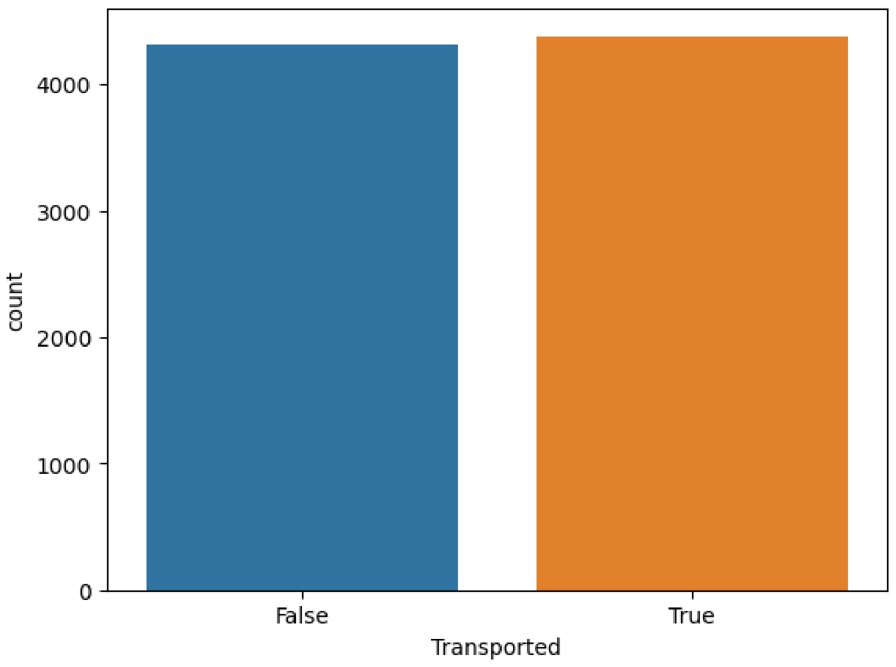
As a result, I got a graph like this:



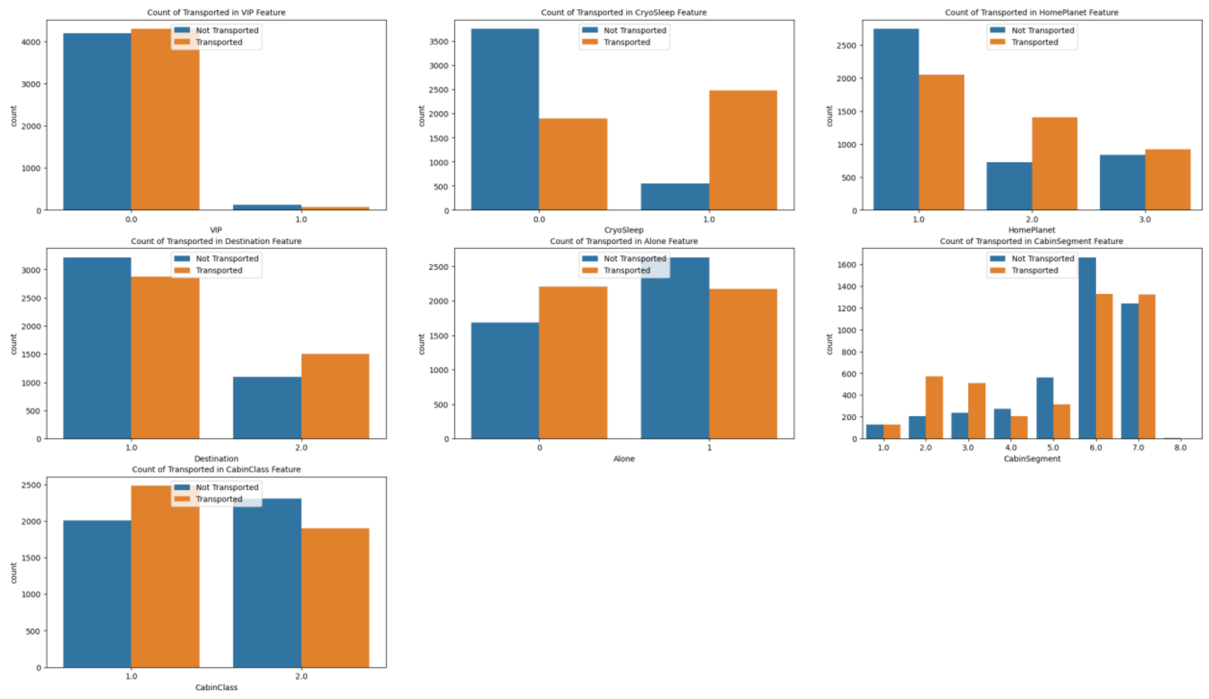
In the graph here, the intensity of the red and green colors indicate the intensity of the relationship. You can find the numerical value of the relationship above the colored boxes.

Other graphs related to the data are shown below:

```
Number of passengers didn't transported = 4315
Number of passengers transported = 4378
```



In the chart above, we see the number of passengers that have been transported and those that have not been transported in our train_csv document.



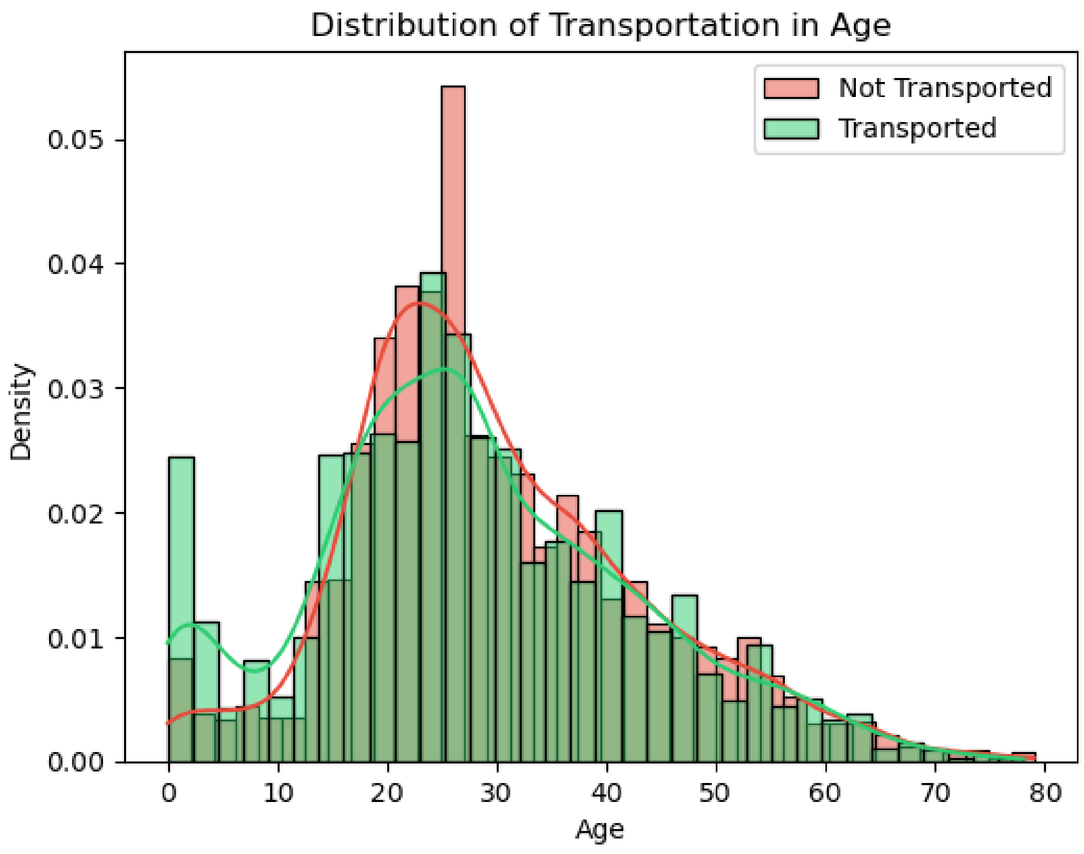
In the chart above, we see the transport status of VIP passengers, passengers in CryoSleep status, according to Home Planet, according to their destination, and whether they are alone or not.

I also discussed the CabinSegment and CabinClass graphics that we created using cabin information. We have a cabin number like this: G/1345/P. Here the letter G means deck and there are types such as A, B, C, D, E, F, G, T. The letter P indicates which side. It can be P or S. P: Port, S: Starboard.

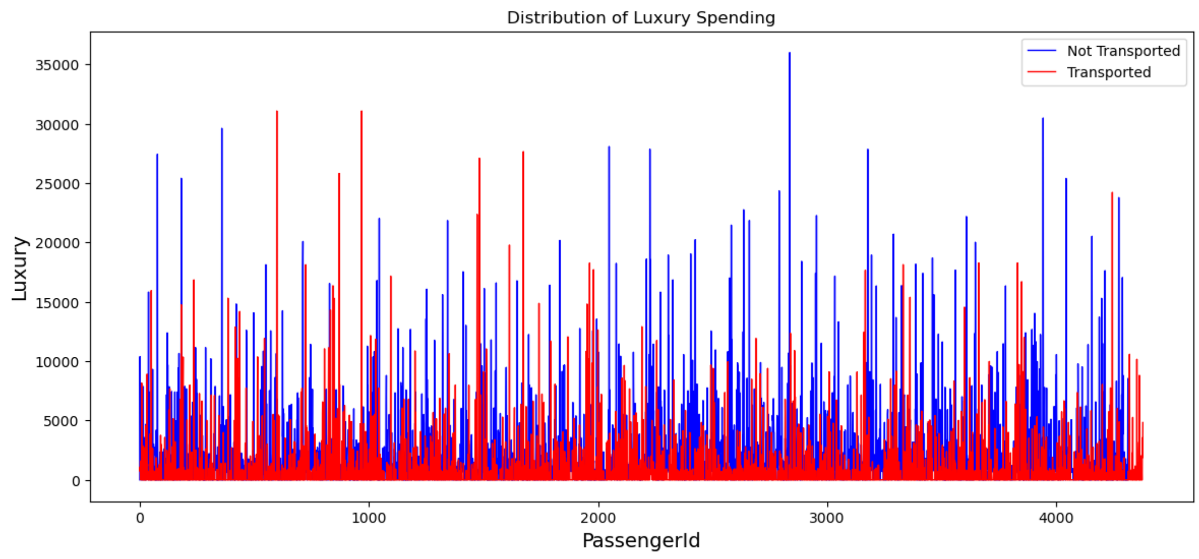
I converted these to numbers. A, B, C, D, E, F, G, T correspond to the numbers 1, 2, 3, 4, 5, 6, 7, 8 respectively. The relevant graph shows how many people were or could not be transported in which of these segments.

Likewise, for the letters P and S, I used the numbers 1 and 2 respectively. The relevant graph shows how many of the items in the Port side and Starboard side sections have been transported or not.

I drew a scatter plot to show the age distribution. In this chart, the green columns show who is transported according to age groups, and the pink columns show who cannot be transported according to age groups.



In the dataset I had (train_csv), there were some column headings that I thought were luxury spending (Room Service, Food Court, Shopping Mall, SPA, VR Deck). I collected all of these in the "Luxury" column. I drew a distribution chart showing the distribution of passengers who were transported and those who were not, according to their expenses.



I noticed in the graphs I drew that the results are quite close to each other. There is a possibility that the model I choose will fail. This could possibly be overfitting. The possibility of an underfitting result actually seemed more attractive.

If we think about it from this perspective, the data set I was interested in was a little intimidating, but the results were much better than I wanted. If we think about it from this perspective, the data set I was interested in was a little intimidating, but the results were much better than I wanted. You will be able to realize this when you see the results and comparison graphs.

6 Feature Engineering

I did most of the Feature Engineering activities while drawing graphics for at least (train_csv). Since I did the same operations for (test_csv), you can consider what I will explain here for both train_csv and test_csv.

I imported the dataset as pandas dataframe. My first goal was to convert string data into numbers as much as possible. After this I thought of filling in non-existent values. I used the SimpleImputer method in the sklearn library to fill in the missing data.

I chose the "median" strategy to fill in data that was not in both the train_csv and test_csv data sets. Actually, there are three different methods that can be used in this library. Since I made almost all of the data numerical, I would use either the mean or median method. However, accuracy values were higher in the median method. That's why I chose median to fill in all missing values. To make my experiments easier, I used a variable called NonStrategy, as can be seen in the notebook.

Transported: I just converted Transported feature to integer.

VIP: I filled in the missing values in the VIP property using SimpleImputer. Since it was already True/False, I didn't need to convert extra.

CryoSleep: I filled in the missing values in the CryoSllep property using SimpleImputer. Since it was already True/False, I didn't need to convert it.

RoomService, FoodCourt, ShoppingMall, Spa, VRDeck: I thought they were luxury expenses for RoomService, FoodCourt, ShoppingMall, Spa, VRDeck features. Therefore, after filling in their missing values, I collected all the features into a feature called *Luxury.*

CabinSegment: In the Cabin feature, the first character indicated which deck the cabin was on. The value here can be A,B,C,D,E,F,G,T. I converted these to numbers and used the numbers 1,2,3,4,5,6,7,8 respectively. I used SimpleImputer to fill in non-existent values. I copied the values I found into a feature called CabinSegment.

CabinClass: In the Cabin feature, the first character showed which side the cabin was on. It could take a value like P or S. I made this value numerical and filled in the missing values. I added all the new values to the dataframe by adding a feature called CabinClass.

Destination: The Destination feature had three destinations. These were TRAPPIST-1e, PSO J318.5-22, 55 Cancri e=3. I defined the same feature as TRAPPIST-1e=1 PSO J318.5-22=2 55 Cancri e=3. I filled in the missing values.

HomePlanet: I converted the HomePlanet feature to numbers, similar to the Destination feature. I looked at the first letter in the Destination feature. I couldn't look at the first letter here. Because the values here are Earth, Europa and Mars, I made the calculation based on the first two letters due to similar first letters. I filled in the missing values.

Age: For the Age feature, I only filled in the missing values.

Alone: The PessengerId feature actually contained information whether a passenger was alone or not. The PessengerId feature actually contained information whether a passenger was alone or not. Passengers with the same first four digits of this number were traveling in groups. I detected these and added a feature called Alone, which carries the values 0 and 1 to distinguish whether alone or not alone. I filled in the missing values.

After completing everything, I deleted the "Cabin", "RoomService", "FoodCourt", "ShoppingMall", "Spa", "VRDeck", "Name", "GroupID", "InGroup" properties that I did not need from the dataframe.

After performing the same operations on both dataframes (train_csv and test_csv), it is time to merge them. After merging, I looked at the types of the fields and a list like the one below emerged. (df dataframe)

```
PassengerId      object
HomePlanet      float64
CryoSleep       float64
Destination     float64
Age             float64
VIP             float64
Transported     float64
Luxury          float64
CabinSegment    float64
CabinClass      float64
Alone             int64
```

From this dataframe, I separated the part containing the data of the train_csv file using the split method. I did the same for test_csv. (dfs[0] dfs[1])

I assigned dfs[0] to the train dataframe and dfs[1] to the test dataframe.

I turned train dataframe into spark dataframe.

7 Implementation of Six Spark ML Models

Since the working time on IBM Cloud was limited, I used the notebook I prepared with the Anaconda and Jupyter Lab application I installed on my own computer. I used Python language for the application

My data set seemed to lend itself to many patterns that I could classify. I can list them as follows:

Logistic Regression

GBT Classifier

Linear SVC

Decision Tree Classifier

Random Forest Classifier

Multilayer Perceptron Classifier

Since I was going to do a accucary analysis for each one of them anyway, I thought of applying them all instead.

For the assignment, I had to use Apache Spark.

Besides the pyspark library, I also used the following libraries:

Os

Pandas

Numpy

Sklearn

Collections

Re

Pylab

Pyspark

Matplotlib

Seaborn

8 Accuracy Benchmark

As you can see from the notebook, I applied six models. You can find their accuracy values opposite.

Model-1: Logistic Regression **0.7369147590014955**

Model-2: GBT Classifier **0.7835039687104567**

Model-3: Linear SVC **0.7367997239157943**

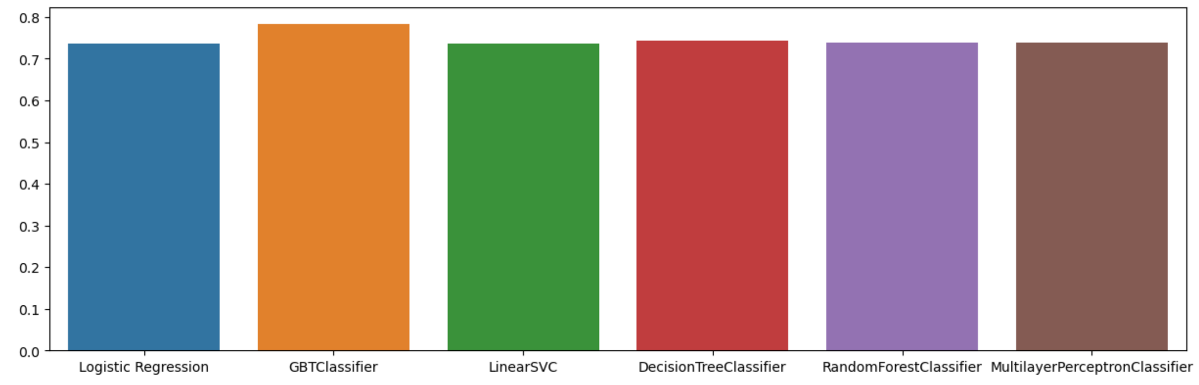
Model-4: Decision Tree Classifier **0.7427815483722535**

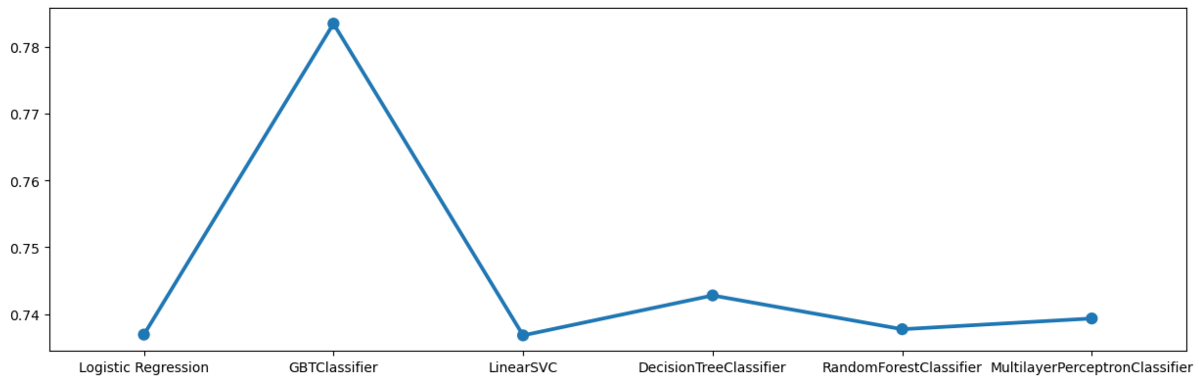
Model-5: Random Forest Classifier **0.7377200046014034**

Model-6: Multilayer Perceptron Classifier **0.7393304958012193**

9 Model Selection and Prediction

The graphics for the models above were created as follows:





"Gradient-Boosted Trees (GBTs)" had the highest accuracy. Therefore, I preferred this model and applied it to test_csv. You can monitor the application results, actual values and predicted values from the notebook.