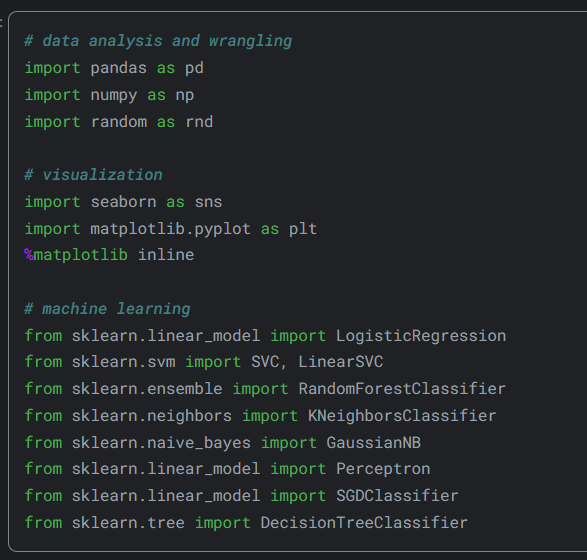
***Solution of MANAV SEHGAL***

**Problem Definition**: Given a dataset with information on passengers who survived or perished in the Titanic disaster, can our model predict whether passengers in a separate test dataset, lacking survival data, survived or not.

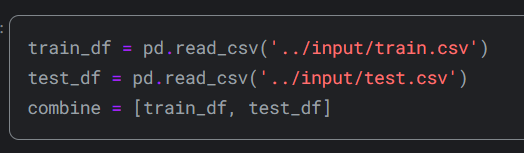
**1. Libraries and Accessing Data**

***a. Libraries***



The necessary libraries were imported.

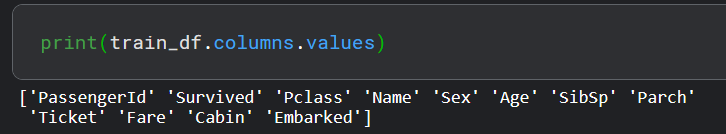
***b.* *Accessing Data***



There are two datasets containing similar passenger information such as name, age, gender, socio-economic class, etc. One is labeled as train.csv and the other as test.csv.Train.csv comprises information on a subset of the passengers (specifically 891 individuals) aboard the Titanic, crucially indicating whether they survived or not, referred to as the 'ground truth'. The test.csv dataset holds analogous information but doesn't provide the 'ground truth' for each passenger. The aim is to forecast these outcomes.

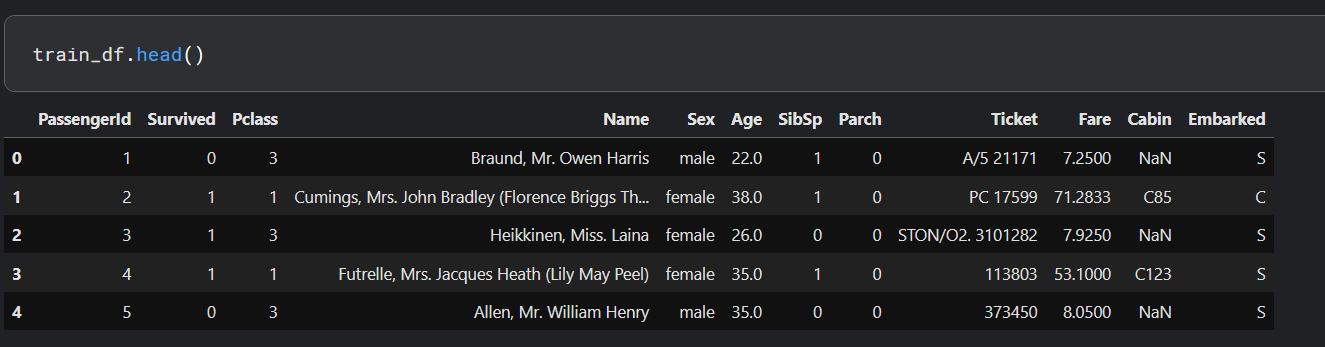
Utilizing the patterns identified in the train.csv data, anticipate the survival status of the remaining 418 passengers aboard (presented in test.csv).

**2. Data Information**

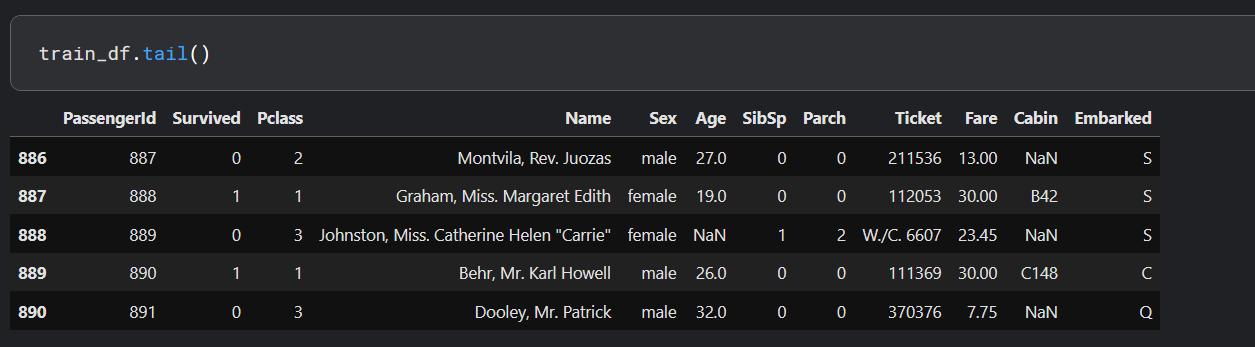


First of all, let's examine the data in our train data. There are 12 different columns in the train dataset. Survived, Sex, Embarked and Class columns are categorical data. In addition, the Pclass column contains categorical ordinal data. The other columns that are Age, Fare, SibSp and Parch are numeric data. Age and Fare are continuous numeric data; SibSp and Parch are discontinuous numeric data.

For example, we can see the first 5 lines in the Picture2.1 and the last 5 lines in the Picture2.2.

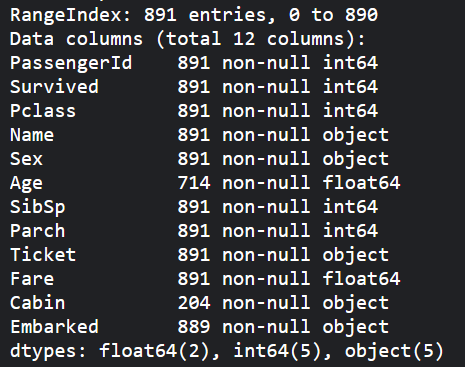


*Picture2.1*



*Picture2.2.*

When we examine the 2 pictures, we see that the ticket and cabin columns are of mixed data type. It contains both numeric and alphanumeric characters. it also turns out that the PassengerId column is an index, and the Name column has a string expression. From a negative point of view, some data are empty or null. We'll have to look into these. We can look at the number of non-empty data when we run the code line that is train\_df.info ().



*Table2.1*

**3. Distribution of Feature Values**

***a.* *Distribution******Numerical Features***

In the solution I examined, Manav Sehgal drew the following conclusions for the numerical features in the data.

• Total samples are 891 or 40% of the actual number of passengers on board the Titanic (2,224).

• Survived is a categorical feature with 0 or 1 values.

• Around 38% samples survived representative of the actual survival rate at 32%.

• Most passengers (> 75%) did not travel with parents or children.

• Nearly 30% of the passengers had siblings and/or spouse aboard.

• Fares varied significantly with few passengers (<1%) paying as high as $512.

• Few elderly passengers (<1%) within age range 65-80.

On the other hand, we can obtain information by using the describe() function.



*Output of train\_df.describe()*

***b.* *Distribution******Categorical Features***

Similarly, Manav Sehgal made some inferences for categorical data. These inferences:

• Names are unique across the dataset (count=unique=891)

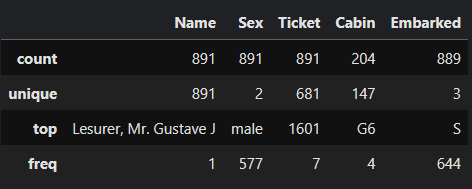
• Sex variable as two possible values with 65% male (top=male, freq=577/count=891).

• Cabin values have several duplicates across samples. Alternatively, several passengers shared a cabin.

• Embarked takes three possible values. S port used by most passengers (top=S)

• Ticket feature has high ratio (22%) of duplicate values (unique=681).

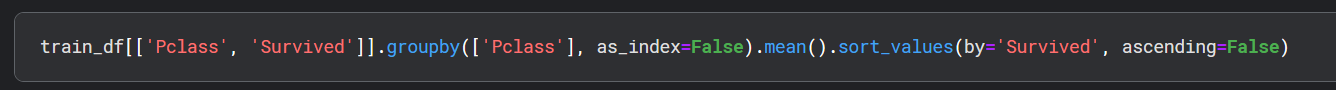
In addition, we can obtain information by using the describe(include=['0']) function.



*Output of train\_df.describe(include=['0'])*

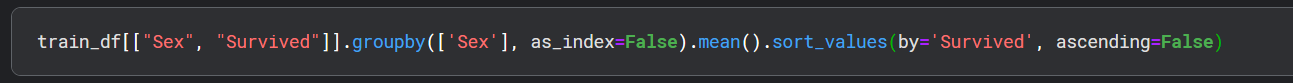
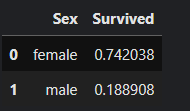
**4. Evaluating Features and Correlation**

To validate certain observations and assumptions, we can promptly examine the correlations between features by pivoting them against each other. However, this analysis is feasible only for features without any missing values. Moreover, it's logical to conduct this analysis solely for features categorized as either categorical (e.g., Sex), ordinal (e.g., Pclass), or discrete (e.g., SibSp, Parch).



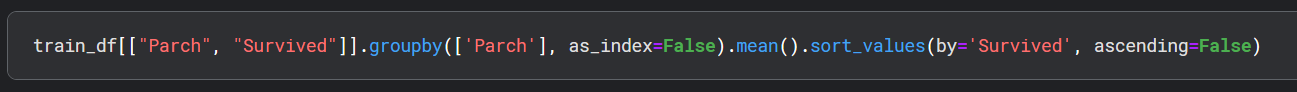
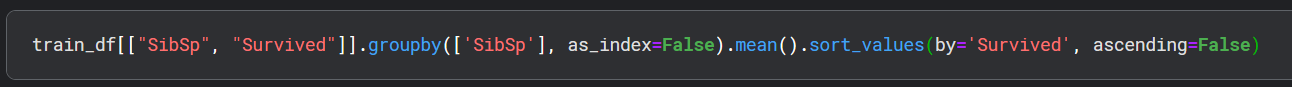
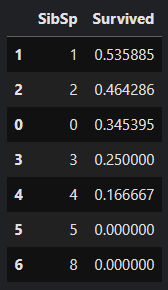
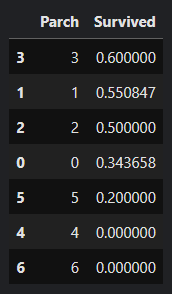
According to the output on the side, we can decipher that there is an important correlation between the "PClass" and the "Survived" columns. The survival average of the "Pclass" column in the data with 1 is 0.629630. On the other hand, where the value is 3, it is 0.242363. In other words, a low value of "Pclass" that is ordinal categorical feature increases the chances of survival.

*Output1*



it is obvious that there is a strong correlation when we look at the correlation relationship between men and women. if we look at the table on the side, it seems that the survival rate of female is 3 times compared to male.

*Output2*

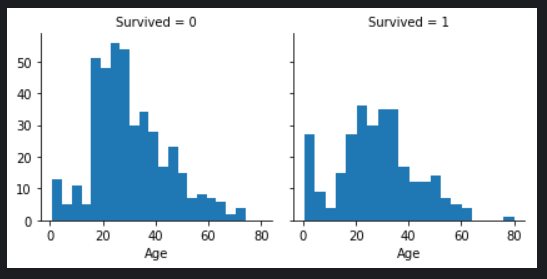
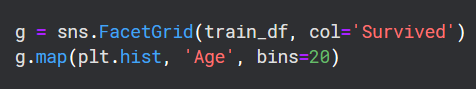


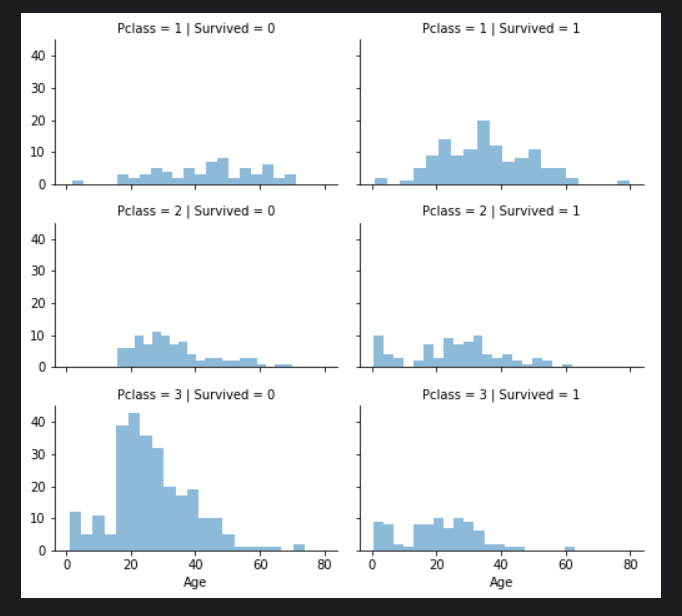
SibSp and Parch features exhibit zero correlation for certain values. It might be optimal to derive a feature or a set of features from these individual attributes. For instance, by combining SibSp and Parch features, we could create a new attribute representing family size. This could offer a more meaningful insight into whether each passenger traveled with their family or not.

*Output3* *Output4*

**5. Visualizing Data**

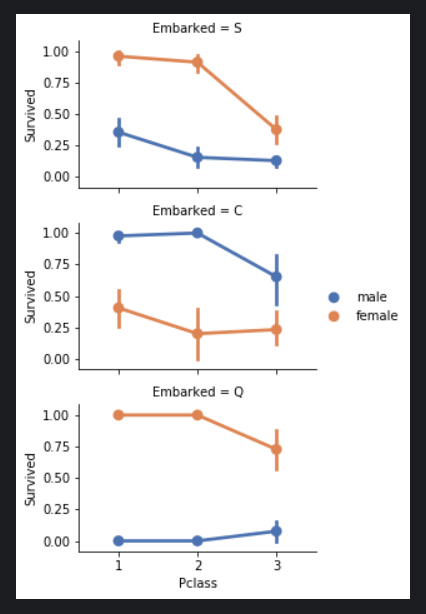
***a.******Correlation of Numerical Features***

 In this part, creating a histogram can be useful for understanding the correlation between numerical features (such as age) and the survival goal. This helps us analyze the distribution of continuous variables like age. Based on our observations, we note that infants (up to 4 years old) had a high survival rate, the oldest passengers (at 80 years old) survived, and a significant number of passengers aged between 15-25 did not survive. This simple analysis confirms that we should consider the age feature in our model training, complete missing values, and create *Graph5.a.1* age groups.

*****b.******Correlation of Numerical and Ordinal Features***

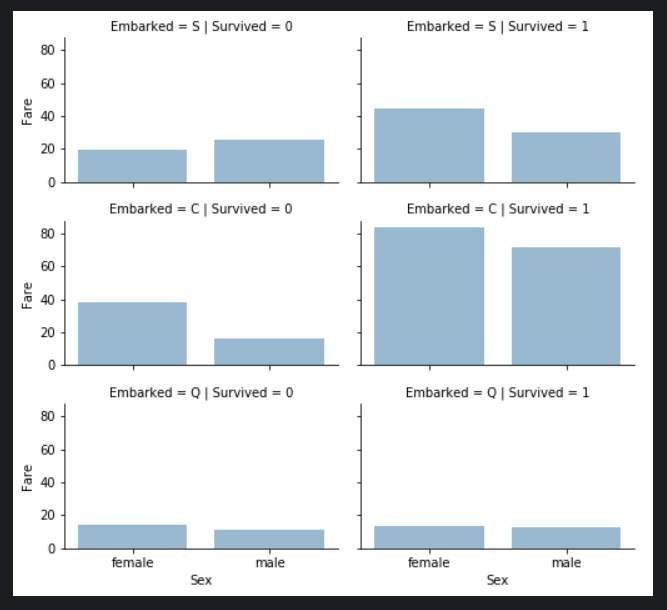
To determine the relationships between numerical and ordinal features we can obtain correlations by combining multiple properties into a single graph. Although Pclass=3 had the most passengers, the majority of them did not survive. The majority of the baby passengers in Pclass=2 and Pclass=3 survived. Most passengers in Pclass =1 survived. These results confirm our classification assumption. Pclass differs in the age distribution of passengers. it is crucial that Consider Pclass for model training. Pclass is an important feature to improve the accuracy of the model. In particular, attention should be paid to the low survival rate of Pclass=3 and the high survival rate of Pclass=1. This information can help the model make more effective predictions.

*Graph5.b.1*

 ***c. Correlation of Categorical Features***

Despite female passengers generally having a higher survival rate compared to males, there is an exception in Embarked=C where males have a higher survival rate, suggesting a potential correlation between Pclass and Embarked, and consequently between Pclass and Survived rather than a direct correlation between Embarked and Survived. Additionally, it is noteworthy that males in Pclass=3 have a better survival rate in ports C and Q compared to Pclass=2. Considering these observations, the model should include the Sex feature and the Embarked feature for training.

*Graph5.c.1*

***d.* *Correlation of categorical and numerical features***

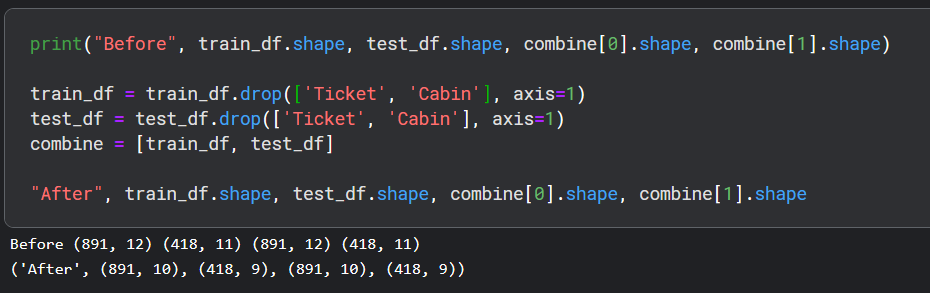
Understanding correlations between categorical non-numeric features and numerical features is crucial. For instance, we may consider correlating Embarked (categorical non-numeric), Sex (categorical non-numeric), and Fare (numerical continuous) features with Survived (categorical numeric). It has been observed that passengers who paid higher fares generally had a higher survival rate, validating the assumption to create fare ranges. Additionally, the port of embarkation seems to be associated with survival rates, supporting the correlation and completion of these features. In light of these observations, it is essential to consider binning the Fare feature for model training. This can enhance the model's *Graph5.d.1* predictive performance by capturing the potential impact of specific fare ranges on survival.

**6. Feature engineering**

Feature engineering involves transforming raw data into informative features to enhance machine learning model performance. It includes tasks like handling missing values, encoding categorical variables, creating new features, and selecting the most relevant ones for modeling. Effective feature engineering improves model accuracy and interpretability, ultimately contributing to the success of machine learning projects.

***a. Dropping Features***

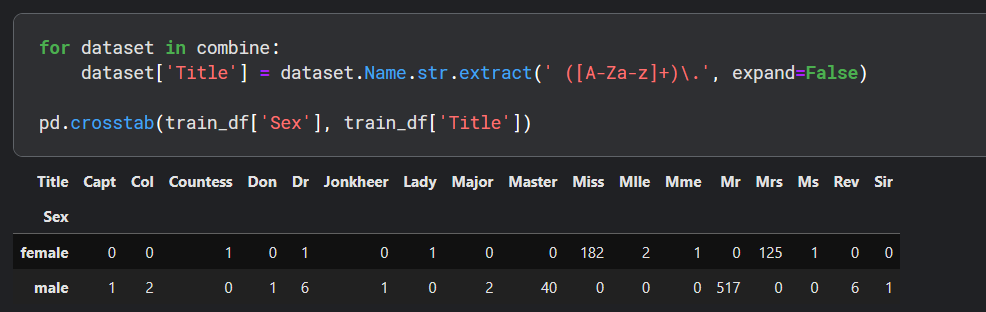
Dropping features is an effective strategy to streamline data analysis by reducing the number of data points and speeding up processing. In accordance with our assumptions and decisions, we aim to drop the Cabin and Ticket features. It's important to note that when applicable, these operations should be applied consistently to both training and testing datasets to maintain coherence.



*Input6.a.1 and Output6.a.1*

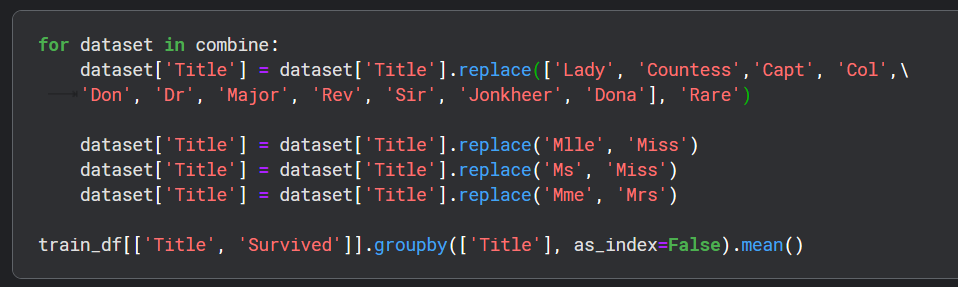
***b. Creating New Feature Extracting from Existing***

Before dropping the Name and PassengerId features, we aim to analyze whether the Name feature can be engineered to extract titles and investigate the correlation between titles and survival. Using regular expressions, we extract the Title feature by matching the first word that ends with a dot character within the Name feature. Observations from plotting Title, Age, and Survived indicate that most titles accurately bandAge groups, with certain titles showing variations in survival rates. For example, titles like Mme, Lady, and Sir mostly survived, while others like Don, Rev, and Jonkheer did not. Based on these observations, we decide to retain the new Title feature for model training***.***



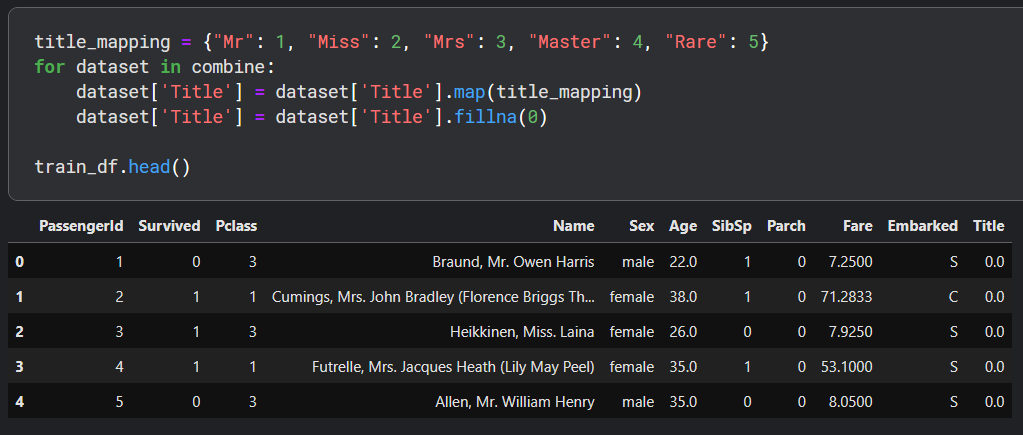
*Input6.b.1 and Output6.b.1*

We have the option to substitute numerous titles with a more frequently occurring name or categorize them as "Rare".



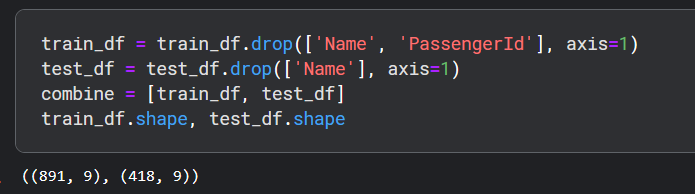
*Input6.b.2*  *Output6.b.2*

We have the possibility to transform the categorical titles into ordinal values.



*Input6.b.3 and Output6.b.3*

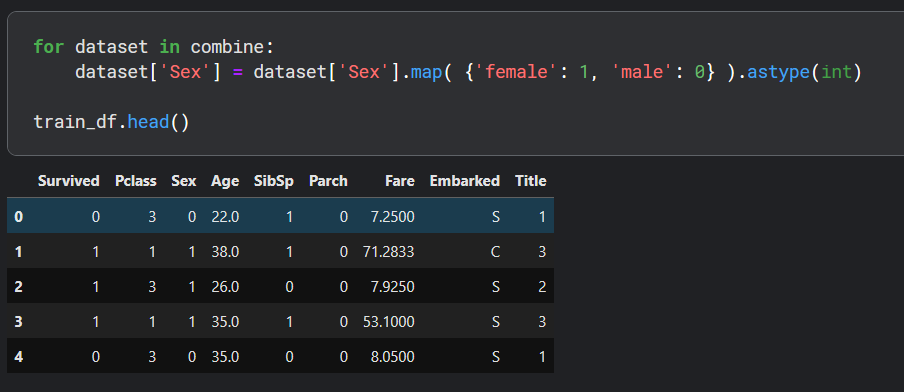
Now, we can confidently remove the Name feature from both the training and testing datasets. Additionally, the PassengerId feature is unnecessary in the training dataset and can also be dropped.

**

*Input6.b.4 and Output6.b.4*

***c.* *Converting Categorical Feature***

We can now proceed to convert string-type features into numerical values, a necessary step for most model algorithms. This conversion will also aid us in fulfilling the goal of completing features. To begin, we'll transform the Sex feature into a new feature named Gender, where female is represented as 1 and male as 0. This conversion simplifies the data representation and enables the algorithm to interpret the information more effectively, contributing to the overall analysis and modeling process.

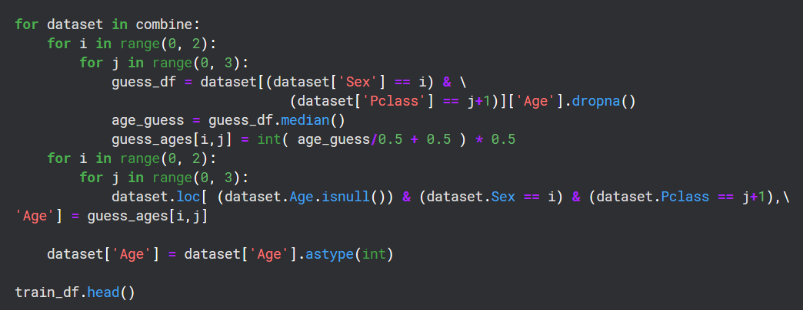


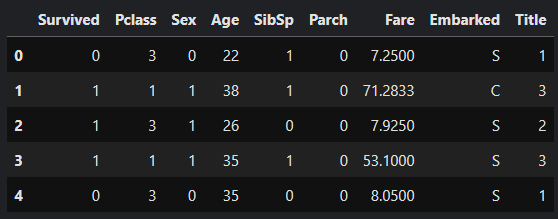
*Input6.c.1 and Output6.c.1*

***d.* *Completing Numerical Continuous Feature***

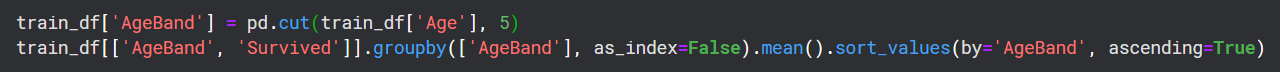
Now, we need to address missing or null values in the Age feature. We have several methods to estimate and complete numerical continuous features. One approach is to generate random numbers between the mean and standard deviation of the existing data. A more accurate method involves using other correlated features. In our case, there's a correlation among Age, Gender, and Class. We can estimate Age values by calculating the median Age for each combination of Pclass and Gender. The third method combines the first two, where instead of using the median, we generate random numbers within the range of the mean and standard deviation, based on Pclass and Gender combinations.

While the first and third methods introduce random noise into our models and may yield varying results with multiple executions, we opt for the second method as it provides more accurate estimates by considering correlations among features. Let's begin by initializing an empty array to store the estimated Age values based on Class x Gender combinations. Next, we'll iterate over Sex (0 or 1) and Pclass (1, 2, 3) to calculate the estimated Age values for the six combinations. Then, we'll create Age bands and determine their correlations with Survived. Finally, we'll replace Age with ordinal values based on these bands, and we drop the AgeBand feature.

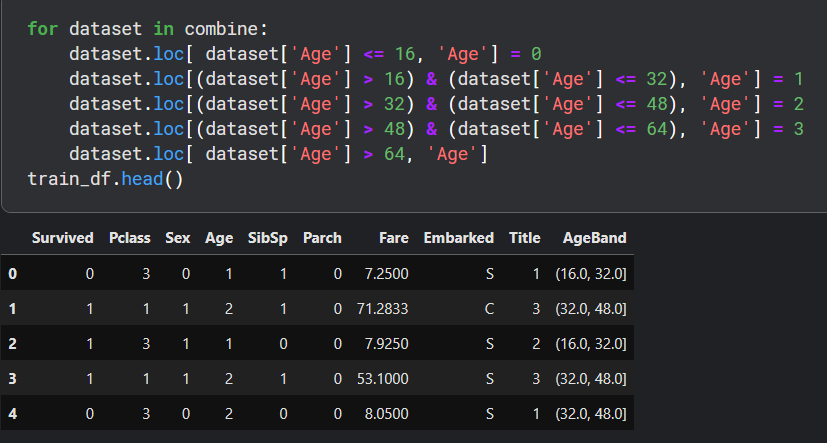
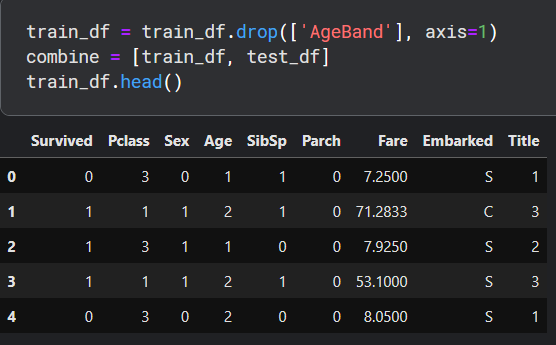




*Input6.d.1 Output6.d.1 Output6.d.2*



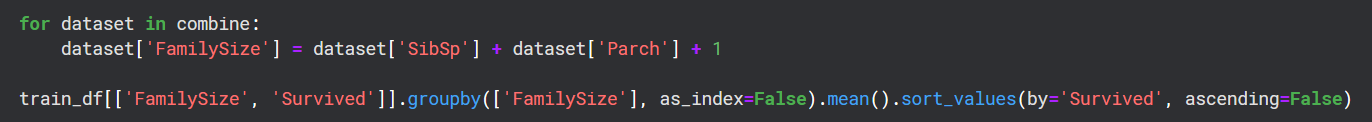
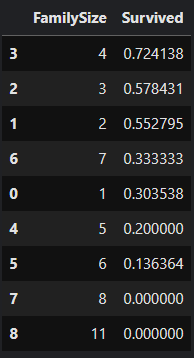
*Input6.d.2*

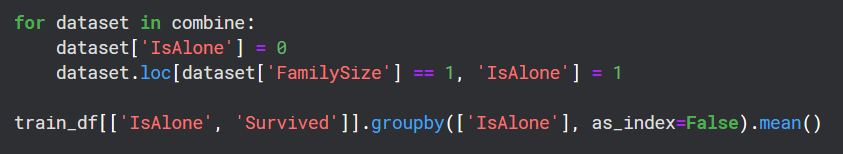
 

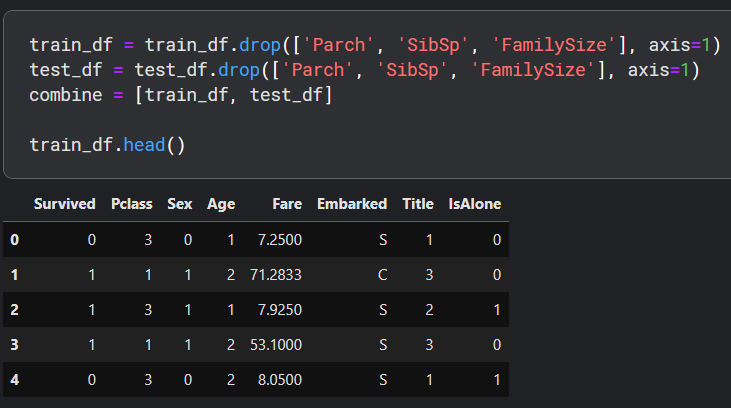
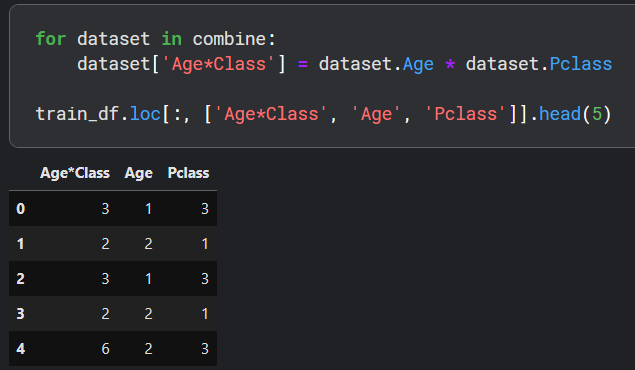
*Input6.d.3 and Output6.d.3 Input6.d.4 and Output6.d.4*

***e.* *Create New Feature Combining Existing Features***

We have the option to generate a new feature called Family Size by combining the Parch and SibSp features, allowing us to discard Parch and SibSp from our datasets. Additionally, we can introduce another feature named IsAlone. By doing so, we can drop Parch, SibSp, and FamilySize features and utilize IsAlone instead. Furthermore, we can create a synthetic feature by combining Pclass and Age. This approach aims to simplify the dataset while retaining relevant information about family size and passenger status, ultimately enhancing the modeling process.



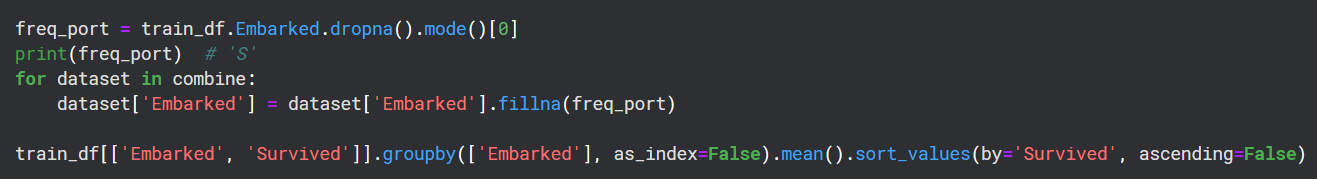
* Input6.e.1*

 *Input6.e.2 Output6.e.2 Output6.e.1*

*Input6.e.3 and Output6.e.3 Input6.e.4 and Output6.e.4*

***f. Completing Categorical Feature***

**The Embarked feature contains values denoting the ports of embarkation, namely S (Southampton), Q (Queenstown), and C (Cherbourg). In our training dataset, there are two missing values in this feature. To address this, we will impute these missing values with the most frequently occurring value, which is the simplest method of handling missing categorical data. This ensures that the dataset remains complete and suitable for analysis without introducing bias or compromising the integrity of the information.



*Input6.f.1 Output6.f.1*

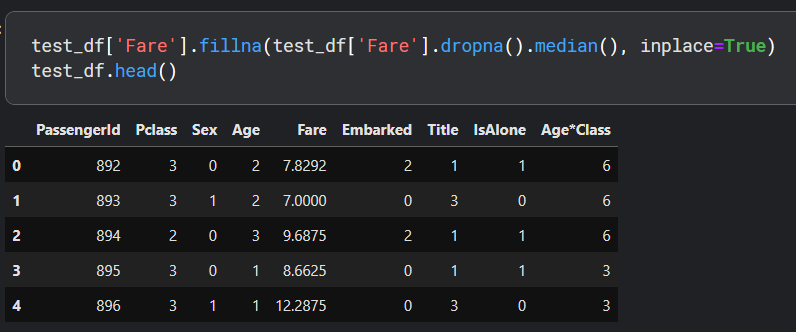
***g. Converting Categorical Feature to Numeric***

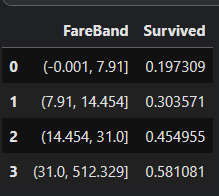
We will convert the Embarked feature into a new numeric Port feature.



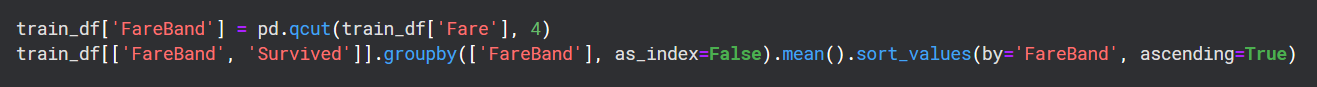
*Input6.g.1 and Output6.g.1*

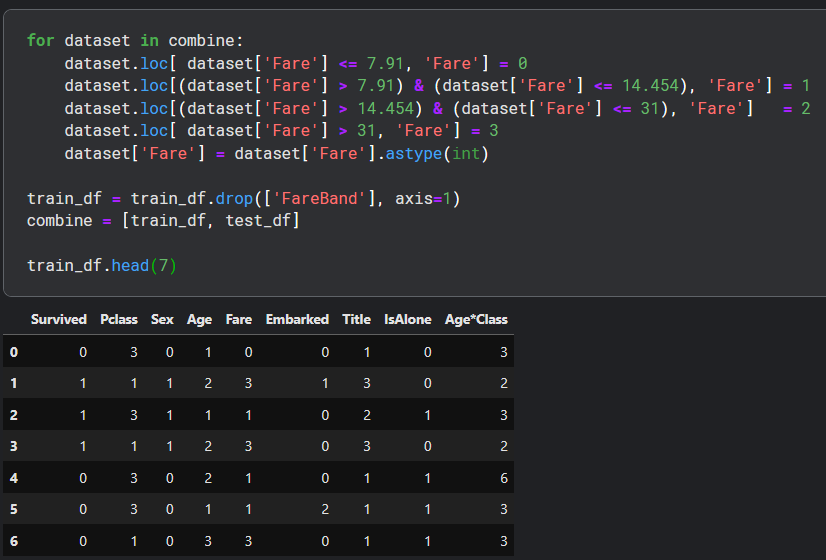
***h. Quick Completing and Converting Numeric Feature***

Now, we can fill in the single missing value in the Fare feature of the test dataset using the model, which represents the value that occurs most frequently for this feature. This process can be accomplished in a single line of code. It's important to note that since we're only replacing a single value, we don't need to create an intermediate new feature or conduct further analysis for correlation to guess the missing feature. This completion goal ensures that the model algorithm operates on non-null values, meeting the desired requirement. Additionally, we may consider rounding off the fare to two decimals to accurately represent currency.



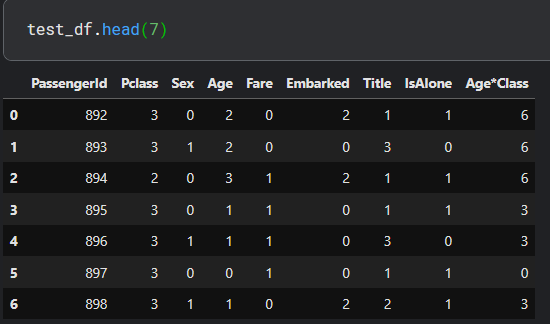
*Input6.h.1 and Output6.h.1 Output6.h.2*

 *Input6.h.2*



*Input6.h.3 and Output6.h.3*

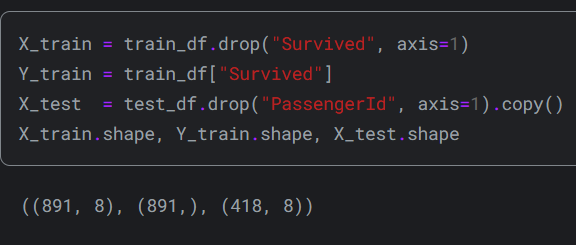
And the first 7 rows of test dataset.



*Input6.h.4 and Output6.h.4*

**7. Model, Predict and Solve**

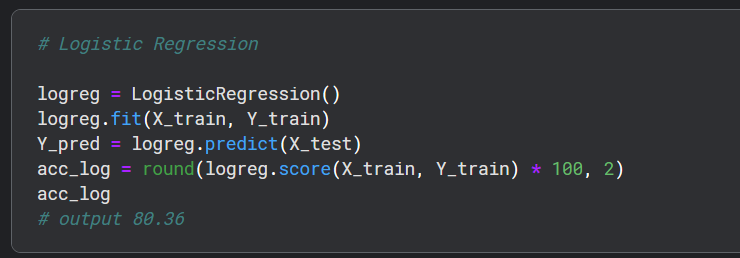
Now that we've completed the data preprocessing steps, we're ready to train a model and make predictions for the required solution. There is a wide range of predictive modeling algorithms to choose from, but we need to consider the type of problem and solution requirements to narrow down our selection to a few models for evaluation. Our problem involves both classification and regression tasks, where we aim to identify the relationship between the output variable (Survived or not) and other input features (such as Gender, Age, and Port). Additionally, since we're training our model with a given dataset, we're dealing with supervised learning. Given these criteria, we can focus our choice of models on a select few. These may include:

* Logistic Regression
* Support Vector Machines
* k-Nearest Neighbors
* Gaussian Naive Bayes
* Perceptron
* Linear SVC
* Stochastic Gradient Descent
* Decision Tree
* Random Forest

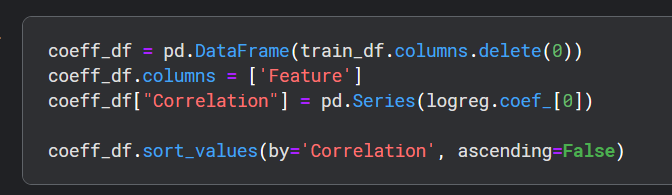
This code snippet is used to separate the features (X\_train) and target variable (Y\_train) for training data by removing the "Survived" column from the training dataframe. Similarly, it selects the features (X\_test) for the test data by dropping the "PassengerId" column from the test data frame. The "shape" function is used to display the dimensions (number of rows and columns) of each data frame. This code segment is crucial for preparing the datasets for machine learning model training, ensuring the proper separation of input features (independent variables) and the target variable (dependent variable).

***a. Logistic Regression***

Logistic Regression is a valuable model to incorporate early in the workflow. It assesses the connection between the categorical dependent variable (feature) and one or more independent variables (features) by estimating probabilities using a logistic function, specifically the cumulative logistic distribution. This method allows us to predict the probability of a binary outcome based on given input features. The confidence score produced by the model, derived from our training dataset, indicates the reliability or certainty of the model's predictions. A higher confidence score suggests greater confidence in the model's predictions, while a lower score indicates less certainty. This score is essential for evaluating the model's performance and its suitability for the given task.



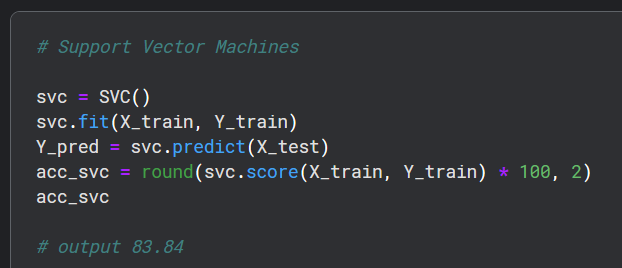
We can utilize Logistic Regression to validate our assumptions and decisions regarding feature creation and completion objectives. This involves calculating the coefficients of the features in the decision function. Positive coefficients indicate an increase in the log-odds of the response (and thus an increase in the probability), while negative coefficients imply a decrease in the log-odds of the response (and thus a decrease in the probability). For example, the highest positive coefficient is associated with the Sex feature, suggesting that as the Sex value increases (from male: 0 to female: 1), the probability of Survived=1 increases the most. Conversely, as Pclass increases, the probability of Survived=1 decreases the most. Additionally, the artificial feature Age\*Class demonstrates the second highest negative correlation with Survived, making it a valuable feature for modeling. Similarly, the Title feature exhibits the second highest positive correlation.

This analysis enables us to understand the impact of each feature on the likelihood of survival, guiding our feature engineering decisions and providing insights into the relationships between features and the target variable.

***b. Support Vector Machines***

We will now employ Support Vector Machines (SVM) for modeling, which are supervised learning models commonly used for classification and regression tasks. SVMs utilize learning algorithms to analyze data and make predictions. When provided with a set of training samples, each labeled as belonging to one of two categories, SVM constructs a model to classify new test samples into one of the categories. SVM is a non-probabilistic binary linear classifier, meaning it assigns data points to one of the categories without providing probability estimates.

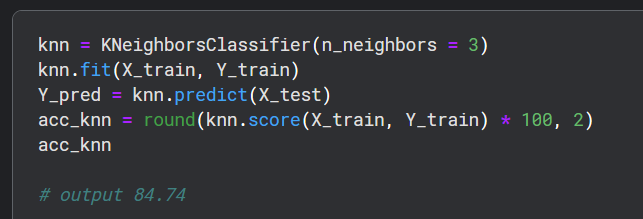
It's worth noting that SVM generates a confidence score, which tends to be higher than that of Logistic Regression models. This higher confidence score indicates greater confidence in the model's predictions, making SVMs particularly useful when aiming for high precision and reliability in classification tasks.



***c. k-Nearest Neighbors***

The k-Nearest Neighbors algorithm (KNN) is a non-parametric method commonly used in pattern recognition for both classification and regression tasks. In k-NN, a sample is classified based on a majority vote of its nearest neighbors, with the sample being assigned to the class most prevalent among its k nearest neighbors. The value of k is a positive integer, typically small. If k equals 1, then the object is simply assigned to the class of its single nearest neighbor.

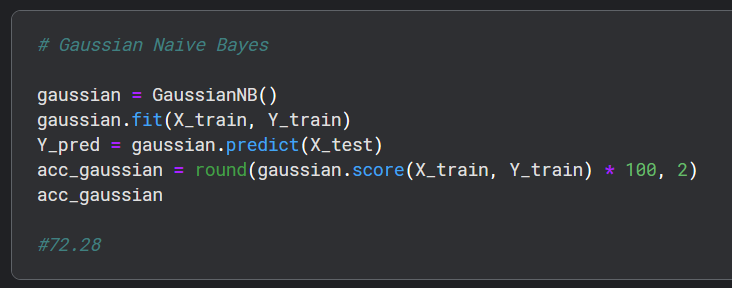
It's important to note that while k-NN generates a confidence score, it tends to perform better than Logistic Regression models but not as well as Support Vector Machines in terms of confidence. This indicates that k-NN may offer intermediate reliability in its predictions compared to the other two models. This algorithm is particularly effective in scenarios where the decision boundaries are complex or nonlinear.



***d. Gaussian Naive Bayes***

Naive Bayes classifiers are a group of straightforward probabilistic classifiers used in machine learning. They operate by applying Bayes' theorem while assuming strong (naive) independence between the features. Naive Bayes classifiers are known for their scalability, as they require a number of parameters that are linear in the number of variables (features) in a learning problem.

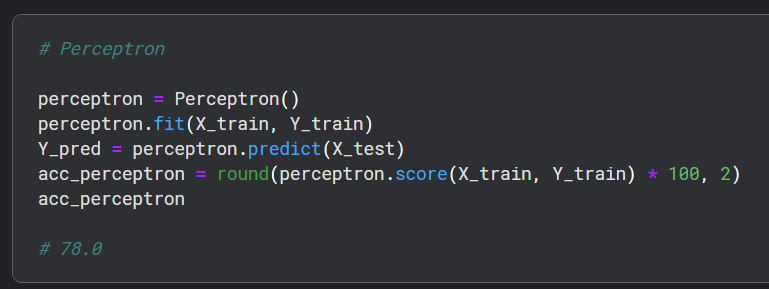
It's noteworthy that while Naive Bayes classifiers offer simplicity and scalability, the confidence score generated by these models tends to be the lowest among the models evaluated thus far. This indicates that Naive Bayes classifiers may provide less certainty or reliability in their predictions compared to other models such as Logistic Regression, Support Vector Machines, and k-Nearest Neighbors. Despite this, Naive Bayes classifiers can still be effective in certain scenarios, particularly when dealing with large datasets and high-dimensional feature spaces.



***e. Perceptron***

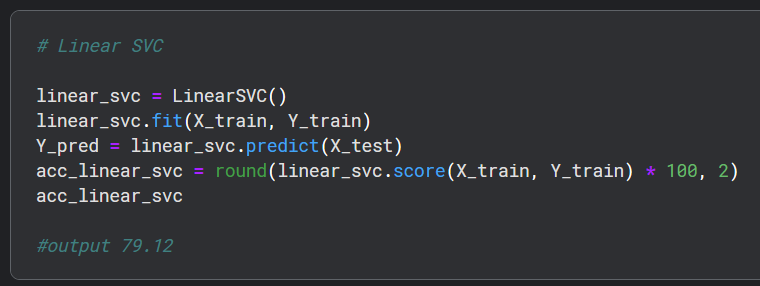
The perceptron is a supervised learning algorithm used to train binary classifiers, which determine whether an input (represented by a vector of numbers) belongs to a specific class or not. It operates as a linear classifier, meaning it makes predictions based on a linear predictor function that combines a set of weights with the feature vector. The perceptron algorithm supports online learning, which means it processes elements in the training set one at a time, allowing for continuous adjustment of the model based on new data.

In essence, the perceptron algorithm iteratively updates its weights to minimize classification errors and optimize its ability to correctly classify input data into the desired categories. This approach makes it particularly useful for tasks involving binary classification and real-time processing, such as spam detection or sentiment analysis.



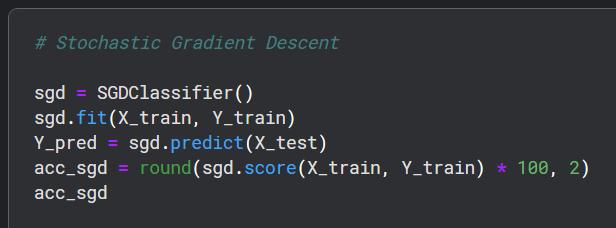
***f. Linear SVC***

Linear SVC is a classification algorithm used to find the optimal linear decision boundary that separates data points into different classes. Unlike other SVM variants, LinearSVC operates directly in the original feature space and is typically suitable for large-scale and high-dimensional datasets. Its main goal is to maximize the margin between classes, aiming to achieve better generalization performance and noise robustness.



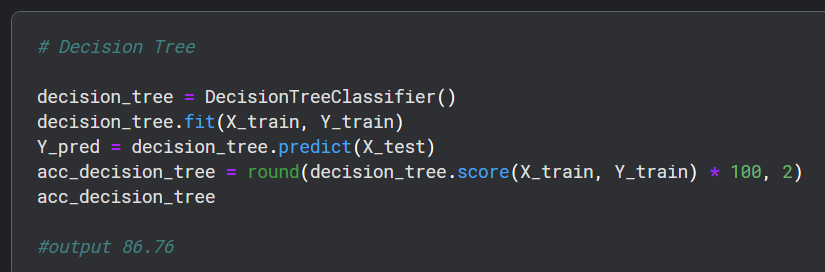
***g. Stochastic Gradient Descent***

Stochastic Gradient Descent (SGD) is an optimization algorithm used to train machine learning models. Instead of computing gradients over the entire dataset, SGD updates model parameters using small batches of data, making it more efficient for large datasets. It iteratively adjusts parameters to minimize a loss function, typically with a predefined learning rate. SGD is widely used due to its speed and scalability, making it suitable for various machine learning tasks.

******

***h. Decision Tree***

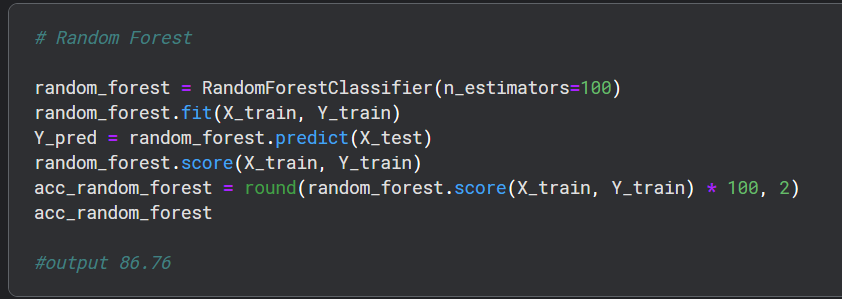
This model utilizes a decision tree as its predictive model, where features (tree branches) are mapped to conclusions about the target value (tree leaves). Decision trees that predict a finite set of values for the target variable are known as classification trees, where leaves represent class labels and branches represent conjunctions of features leading to those labels. On the other hand, decision trees predicting continuous values for the target variable are called regression trees. The confidence score of this model is the highest among the models evaluated so far. Decision trees are commonly used for their simplicity in understanding the dataset and providing high interpretability.



***i. Random Forest***

The Random Forests model is widely regarded as one of the most popular machine learning algorithms. It belongs to the ensemble learning methods category, which combines multiple models to improve predictive performance. Random forests work by constructing a large number of decision trees during training (typically 100 trees) and aggregating their predictions to make the final decision. For classification tasks, the class that occurs most frequently among the individual trees is selected as the output, while for regression tasks, the mean prediction of all trees is taken.

The confidence score of this model is the highest among the models evaluated so far, indicating a high level of confidence in its predictions. Therefore, we have decided to use the output (Y\_pred) of this model for creating our competition submission of results. Random forests are favored for their robustness, versatility, and ability to handle high-dimensional data, making them a popular choice for various classification and regression tasks in machine learning.



**8. Model Evaluation**

Now, we can assess and rank all the models to determine the best one for our problem. Although both the Decision Tree and Random Forest models achieve the same score, we opt to utilize the **Random Forest** model. The reason behind this decision is that Random Forests address the tendency of decision trees to overfit the training set. By constructing multiple decision trees and aggregating their predictions, Random Forests offer improved generalization performance and robustness against overfitting. Therefore, despite scoring equally with the Decision Tree model, we choose Random Forests for their ability to mitigate the overfitting issue commonly associated with decision trees.

