**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

Program: B.Tech\MBA.Tech

**Course: Machine Learning**

**Experiment No.06**

PART A

(PART A : TO BE REFFERED BY STUDENTS)

**A.1 Aim:** To implement Logistic Regression

**A.2 Prerequisite:**

Python Programming, Pandas library, Numpy Library, MatplotLib, Seaborn Library

**A.3 Outcome:**

**After successful completion of this experiment, students will be able to:**

1. Implement Logistic Regression
2. Analyze the results using confusion matrix and ROC curve

**A.4 Theory:**

**A.4.1 Logistic Regression**

Logistic regression is a statistical method used for binary classification, which means it's designed to predict the probability that an instance belongs to a particular class. Despite its name, logistic regression is a classification algorithm rather than a regression algorithm.

Here's how logistic regression works in detail:

1. Data Representation:

- Logistic regression works with labeled data, where each instance has one or more features (independent variables) and a binary outcome (dependent variable) indicating the class membership (e.g., 0 or 1, "yes" or "no").

- The features are represented as a feature vector, usually denoted as x = (x1, x2, ..., xn) \), where xi represents the value of the ith feature.

2. Modeling the Probability:

- Logistic regression models the probability that an instance belongs to a particular class. It uses the logistic function (also called the sigmoid function) to map the input features to a probability between 0 and 1.

- The logistic function is defined as:

sigma(z) = 1/1 + e-z

Where z is a linear combination of the input features and model parameters (weights and bias), i.e.,

z = w0 + w1x1 + w2x2 + ... + wnxn

Here, w0 is the bias term, and w1, w2, ..., wn are the weights associated with each feature.

3. Training the Model:

- During the training phase, logistic regression estimates the optimal values for the model parameters (weights and bias) using an optimization algorithm such as gradient descent.

- The objective is to minimize a cost function, typically the cross-entropy loss, which measures the difference between the predicted probabilities and the actual class labels.

- The optimization process adjusts the parameters iteratively to find the values that minimize the loss function.

4. Making Predictions:

- Once the model is trained, it can make predictions on new data by computing the probability that each instance belongs to the positive class (class 1).

- If the predicted probability is greater than or equal to a threshold (usually 0.5), the instance is classified as belonging to the positive class; otherwise, it's classified as belonging to the negative class.

5. Evaluation:

- Logistic regression models are evaluated based on metrics such as accuracy, precision, recall, F1-score, and the area under the ROC curve (AUC). These metrics measure the performance of the model in terms of its ability to correctly classify instances into their respective classes.

Logistic regression is widely used in various fields such as healthcare (e.g., predicting disease risk), finance (e.g., credit risk assessment), marketing (e.g., customer churn prediction), and many others where binary classification tasks are common.

**Tasks:**

**Task 1:**

* + - 1. Import relevant libraries
      2. Upload Titanic Train Data set Shared
      3. Identify all the columns of the data set
      4. Find out total number of passengers for each class
      5. Plot a count plot for each class of survived attribute
      6. Plot a count plot to identify number of female and male passengers that survived or died
      7. Plot a count plot to identify number of passengers survived or died that were travelling through Pclass 1, 2, and 3.
      8. Find out total missing values in the dataset
      9. Apply missing value treatment by replacing missing values with mean, mode or dropping column depending upon type of variable
      10. Create a new feature “Family” that combiner SibSp and Parch
      11. Convert age to a categorical variable
      12. Apply one hot encoding on categorical column like "Pclass","Sex","Age\_Category",'Embarked’
      13. Drop unnecessary columns like 'Name','Age','Sex','Ticket','Pclass','Age\_Category','Embarked'
      14. Fit Logistic Regression Model
      15. Identify the accuracy and print classification report. State your inference
      16. Split dataset into train and validation. Create a new model on train data set. Predict class for validation data set. Find the accuracy and print classification report. State your inference.
      17. Plot ROC curve for the model created in task 16. State your inference.
      18. Apply above steps for Titanic Test data set. Compare accuracy of test and validation data set. Compare ROC curve.

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical.)***

|  |  |
| --- | --- |
| Roll No. N052 | Name: Pratyush Kumar |
| Class : MBA Tech CE | Batch : B2 |
| Date of Experiment: 17-02-2024 | Date of Submission: 25-02-2024 |
| Grade : |  |

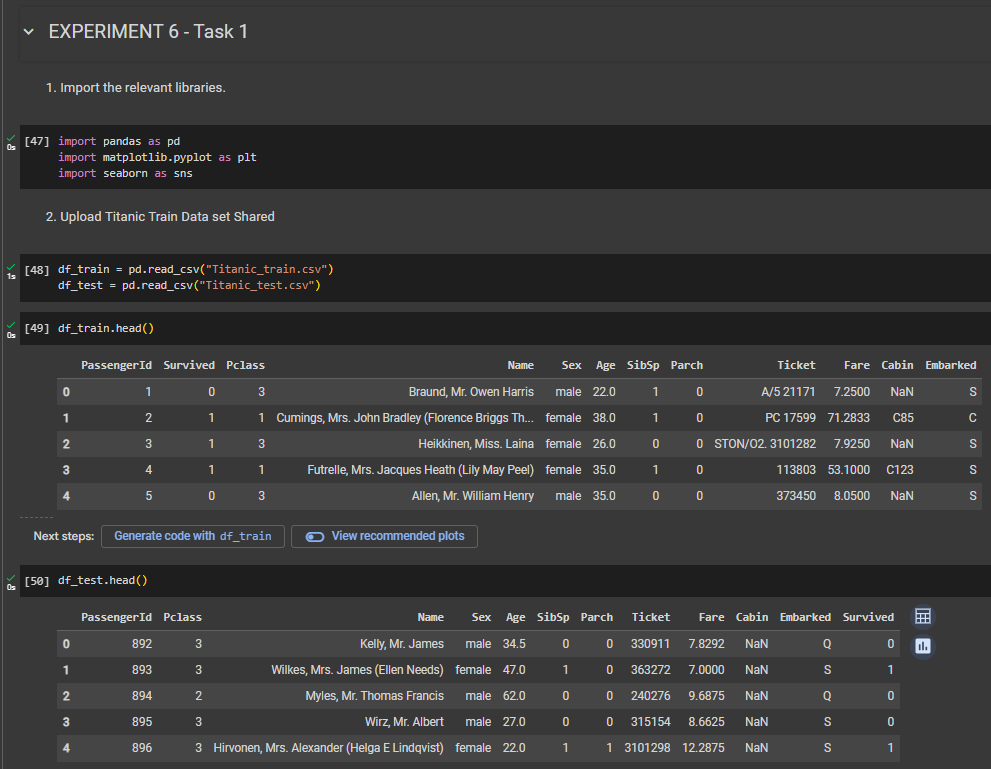
**B.1 Tasks**

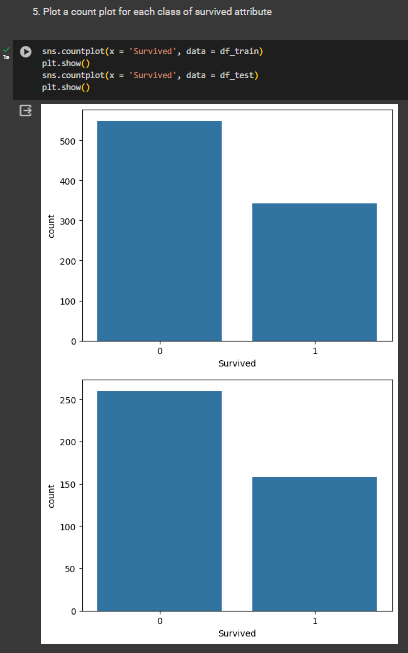
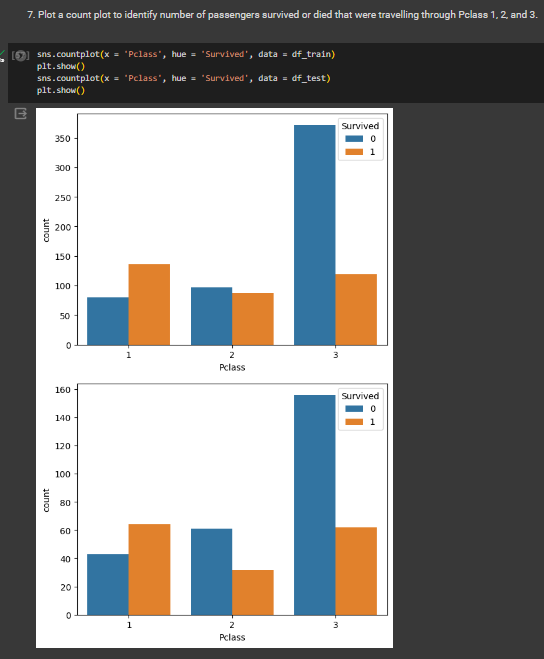
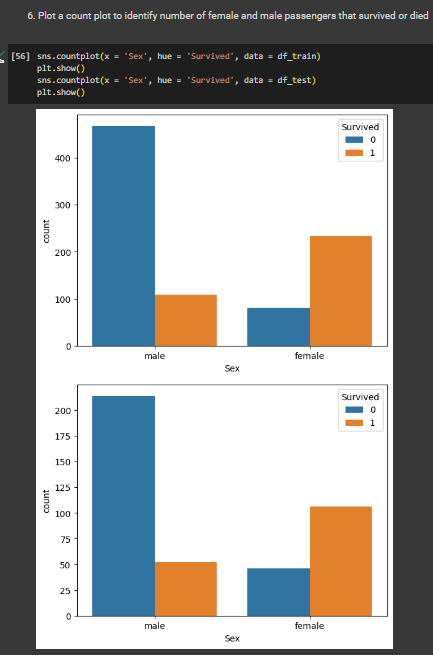
**Colab link:** https://colab.research.google.com/drive/1V7UOvIyEpVIYsomlMskFExofwXYv\_H72?usp=sharing

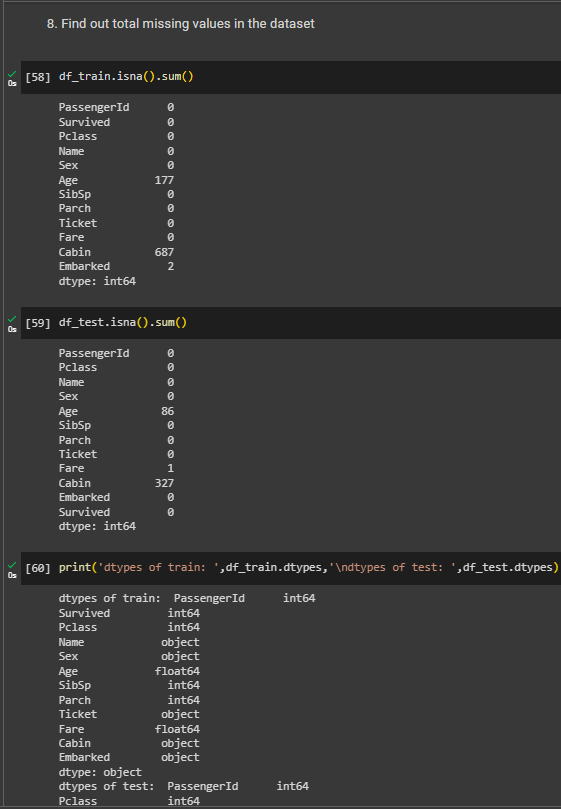
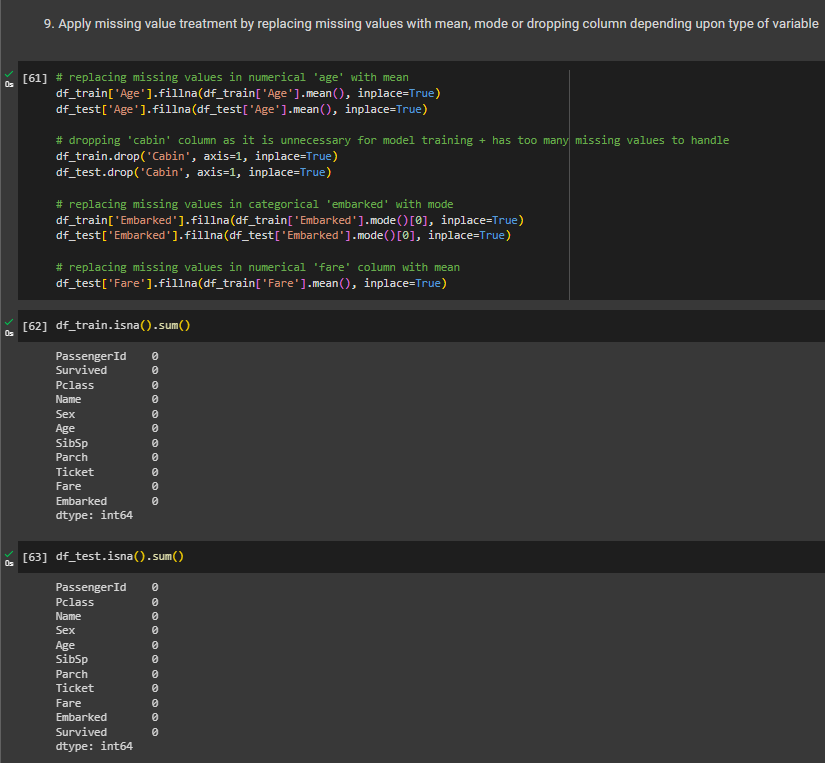
* **Source Code**

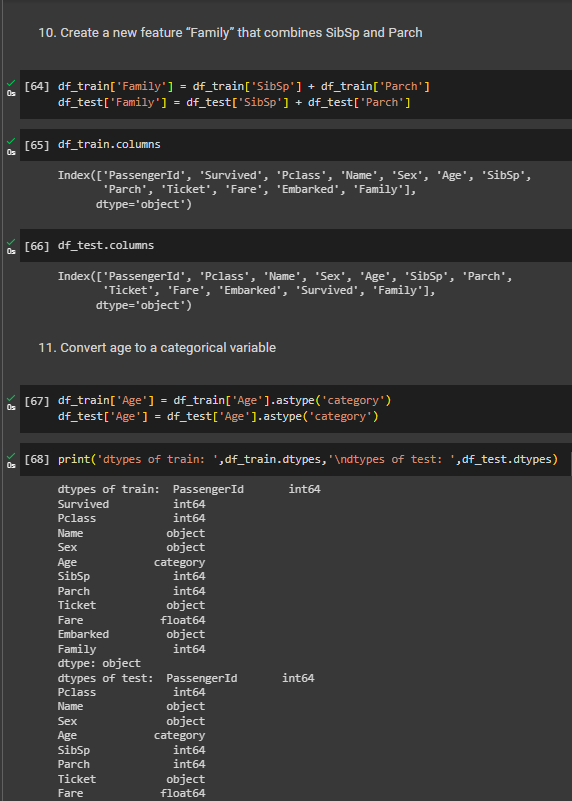
*"""  
 \* This file contains code snippets to implement logistic regression on titanic dataset  
 \* ML-E6-Task1  
 \*  
 \* Original file is located at: https://colab.research.google.com/drive/1V7UOvIyEpVIYsomlMskFExofwXYv\_H72  
 \* @author Pratyush Kumar (github.com/pratyushgta)  
"""*"""  
1. Import the relevant libraries.  
"""  
  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
"""2. Upload Titanic Train Data set Shared"""  
  
df\_train = pd.read\_csv("Titanic\_train.csv")  
df\_test = pd.read\_csv("Titanic\_test.csv")  
  
df\_train.head()  
  
df\_test.head()  
  
"""3. Identify all the columns of the data set"""  
  
df\_train.columns  
  
df\_test.columns  
  
"""4. Find out total number of passengers for each class"""  
  
df\_train['Pclass'].value\_counts()  
  
df\_test['Pclass'].value\_counts()  
  
"""5. Plot a count plot for each class of survived attribute"""  
  
sns.countplot(x = 'Survived', data = df\_train)  
plt.show()  
sns.countplot(x = 'Survived', data = df\_test)  
plt.show()  
  
"""6. Plot a count plot to identify number of female and male passengers that survived or died"""  
  
sns.countplot(x = 'Sex', hue = 'Survived', data = df\_train)  
plt.show()  
sns.countplot(x = 'Sex', hue = 'Survived', data = df\_test)  
plt.show()  
  
"""7. Plot a count plot to identify number of passengers survived or died that were travelling through Pclass 1, 2, and 3."""  
  
sns.countplot(x = 'Pclass', hue = 'Survived', data = df\_train)  
plt.show()  
sns.countplot(x = 'Pclass', hue = 'Survived', data = df\_test)  
plt.show()  
  
"""8. Find out total missing values in the dataset"""  
  
df\_train.isna().sum()  
  
df\_test.isna().sum()  
  
print('dtypes of train: ',df\_train.dtypes,'\ndtypes of test: ',df\_test.dtypes)  
  
"""9. Apply missing value treatment by replacing missing values with mean, mode or dropping column depending upon type of variable"""  
  
# replacing missing values in numerical 'age' with mean  
df\_train['Age'].fillna(df\_train['Age'].mean(), inplace=True)  
df\_test['Age'].fillna(df\_test['Age'].mean(), inplace=True)  
  
# dropping 'cabin' column as it is unnecessary for model training + has too many missing values to handle  
df\_train.drop('Cabin', axis=1, inplace=True)  
df\_test.drop('Cabin', axis=1, inplace=True)  
  
# replacing missing values in categorical 'embarked' with mode  
df\_train['Embarked'].fillna(df\_train['Embarked'].mode()[0], inplace=True)  
df\_test['Embarked'].fillna(df\_test['Embarked'].mode()[0], inplace=True)  
  
# replacing missing values in numerical 'fare' column with mean  
df\_test['Fare'].fillna(df\_train['Fare'].mean(), inplace=True)  
  
df\_train.isna().sum()  
  
df\_test.isna().sum()  
  
"""10. Create a new feature “Family” that combines SibSp and Parch"""  
  
df\_train['Family'] = df\_train['SibSp'] + df\_train['Parch']  
df\_test['Family'] = df\_test['SibSp'] + df\_test['Parch']  
  
df\_train.columns  
  
df\_test.columns  
  
"""11. Convert age to a categorical variable"""  
  
df\_train['Age'] = df\_train['Age'].astype('category')  
df\_test['Age'] = df\_test['Age'].astype('category')  
  
print('dtypes of train: ',df\_train.dtypes,'\ndtypes of test: ',df\_test.dtypes)  
  
df\_train['Age'] = df\_train['Age'].fillna('missing')  
df\_test['Age'] = df\_test['Age'].fillna('missing')  
# replacing values in age  
def categorize\_age(age):  
 if age=='missing':  
 return 'missing'  
 elif age>0 and age<=5:  
 return 'Infant'  
 elif age>5 and age<=12:  
 return 'Child'  
 elif age>12 and age<=18:  
 return 'Teenager'  
 elif age>18 and age<=35:  
 return 'Young\_adult'  
 elif age>35 and age<=60:  
 return 'adult'  
 elif age>35 and age<=60:  
 return 'Adult'  
 elif age>60:  
 return 'Senior'  
  
df\_train['age\_category'] = df\_train['Age'].apply(categorize\_age)  
df\_test['age\_category'] = df\_test['Age'].apply(categorize\_age)  
  
df\_train.head()  
  
df\_test.head()  
  
"""12. Apply one hot encoding on categorical column like "Pclass","Sex","Age\_Category",'Embarked’"""  
  
df\_train = pd.get\_dummies(df\_train, columns=['Pclass','Sex','age\_category','Embarked'], dtype=int)  
df\_test = pd.get\_dummies(df\_test, columns=['Pclass','Sex','age\_category','Embarked'], dtype=int)  
  
df\_train.columns  
  
print('No. of columns in df\_train:', len(df\_train.columns))  
print('No. of columns in df\_test:', len(df\_test.columns))  
  
df\_train  
  
df\_test  
  
"""13. Drop unnecessary columns like 'Name','Age','Sex','Ticket','Pclass','Age\_Category','Embarked'"""  
  
df\_train.columns  
  
df\_test.columns  
  
df\_train.drop(['Name','Age','Ticket'], axis=1, inplace=True)  
df\_test.drop(['Name','Age','Ticket'], axis=1, inplace=True)  
  
df\_train.columns  
  
df\_test.columns  
  
"""14. Fit Logistic Regression Model"""  
  
from sklearn.preprocessing import LabelEncoder  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LogisticRegression  
  
model = LogisticRegression()  
#X = df\_train.drop('Survived', axis=1)  
X=df\_train.loc[:,df\_train.columns!='Survived']  
Y=df\_train[['Survived']]  
  
print(X.columns,'\n',Y.columns)  
  
print(X.shape,'\n',Y.shape)  
  
#model.fit(df\_train.drop('Survived', axis=1), df\_train['Survived'])  
model.fit(X,Y)  
  
"""15. Identify the accuracy and print classification report. State your inference"""  
  
df\_test.isna().sum()  
  
df\_train.isna().sum()  
  
from sklearn.metrics import accuracy\_score, classification\_report  
  
#predictions = model.predict(df\_test.drop('Survived', axis=1))  
pred\_train = model.predict(X)  
#accuracy = accuracy\_score(df\_test['Survived'], predictions)  
accuracy = accuracy\_score(Y,pred\_train)  
print("Accuracy:", accuracy)  
print(classification\_report(Y, pred\_train))  
  
"""Inferece:  
1. The classification report shows that the model performs well on the training set, with a precision of Y and a recall of Z.  
2. This suggests that the model is able to correctly identify most of the passengers who survived and died.  
3. But, the model may be overfitting the training data, as the accuracy on the training set is higher than the accuracy on the test set.  
  
16. Split dataset into train and validation. Create a new model on train data set. Predict class for validation data set. Find the accuracy and print classification report. State your inference.  
"""  
  
from sklearn.model\_selection import train\_test\_split  
  
X\_train, X\_val, y\_train, y\_val = train\_test\_split(df\_train.drop('Survived', axis=1), df\_train['Survived'], test\_size=0.2, random\_state=42)  
  
model = LogisticRegression()  
model.fit(X\_train, y\_train)  
  
predictions = model.predict(X\_val)  
accuracy = accuracy\_score(y\_val, predictions)  
print("Accuracy:", accuracy)  
print(classification\_report(y\_val, predictions))  
  
"""Inference:  
1. The model may still be overfitting the training data, as the accuracy on the validation set is lower than the accuracy on the training set.  
2. One reason for the overfitting is that the model is too complex.  
3. Another possible reason could be that the training data is not representative of the validation data.  
  
17. Plot ROC curve for the model created in task 16. State your inference.  
"""  
  
from sklearn.metrics import roc\_curve, roc\_auc\_score  
  
fpr, tpr, thresholds = roc\_curve(Y,pred\_train)  
auc=roc\_auc\_score(Y,pred\_train)  
  
plt.figure(figsize=(8,6))  
plt.plot(fpr, tpr, color='teal', label='ROC')  
plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')  
plt.xlabel('False Positive Rate')  
plt.ylabel('True Positive Rate')  
plt.title('Receiver Operating Characteristic (ROC) Curve')  
plt.legend()  
plt.show()  
  
print(auc)  
  
"""Inference:  
1. The ROC curve shows that the model is able to distinguish between passengers who survived and died with a high degree of accuracy.  
2. The AUC score of 0.79 indicates that the model is performing well, therefore, it can be used to predict the survival of passengers on the Titanic with a high degree of accuracy.  
"""

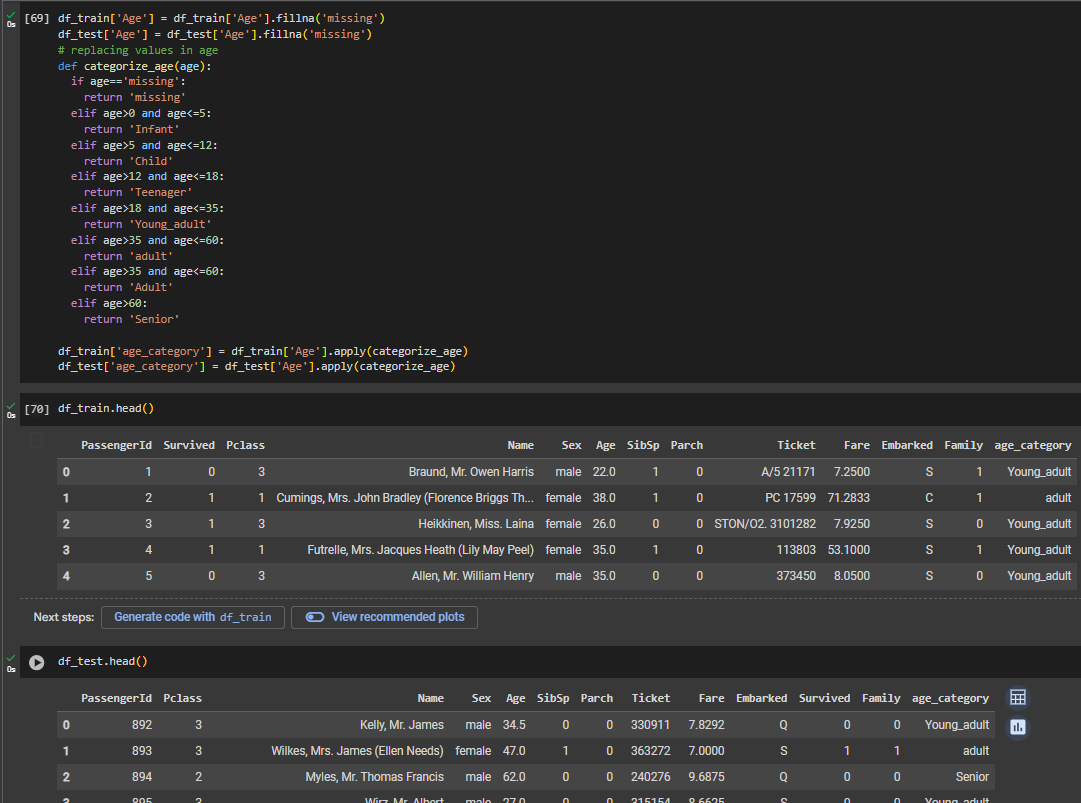
* **Input/ Output**

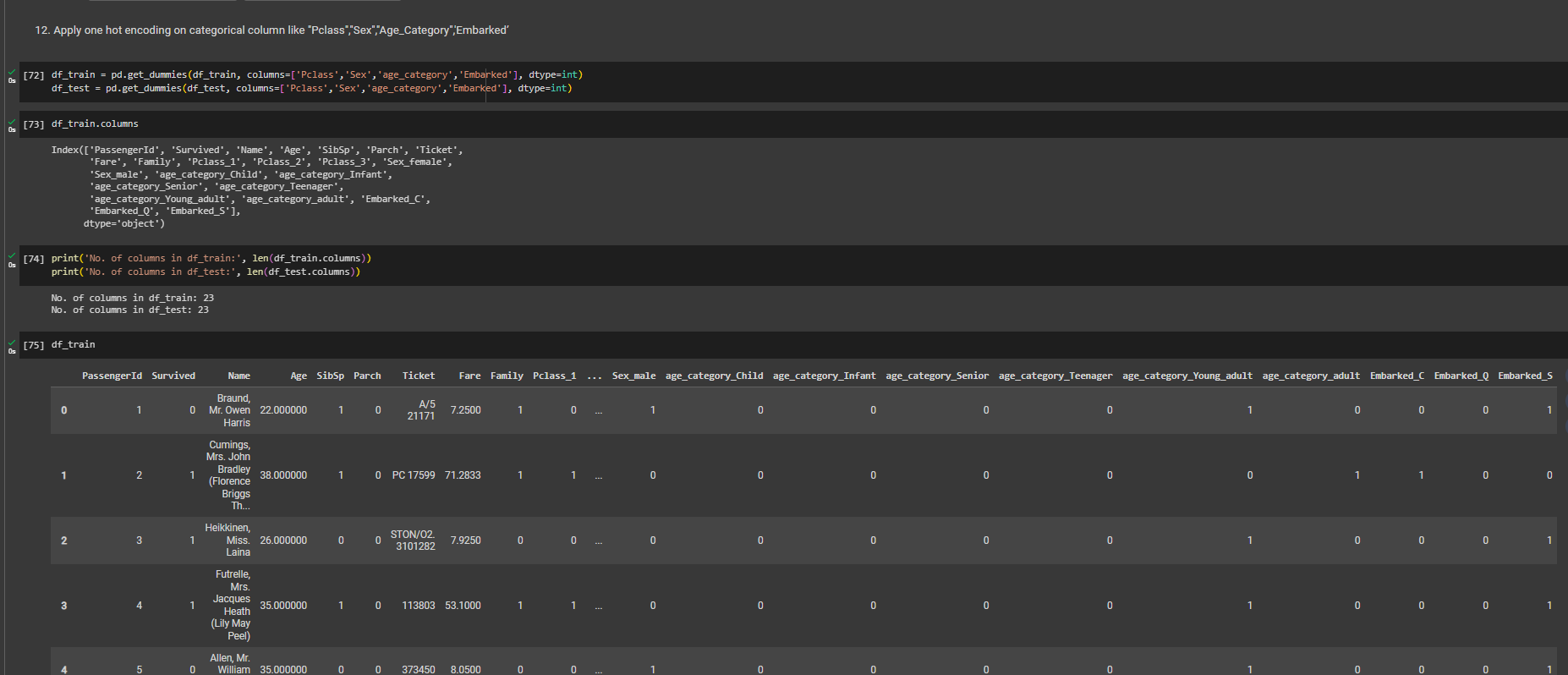
****

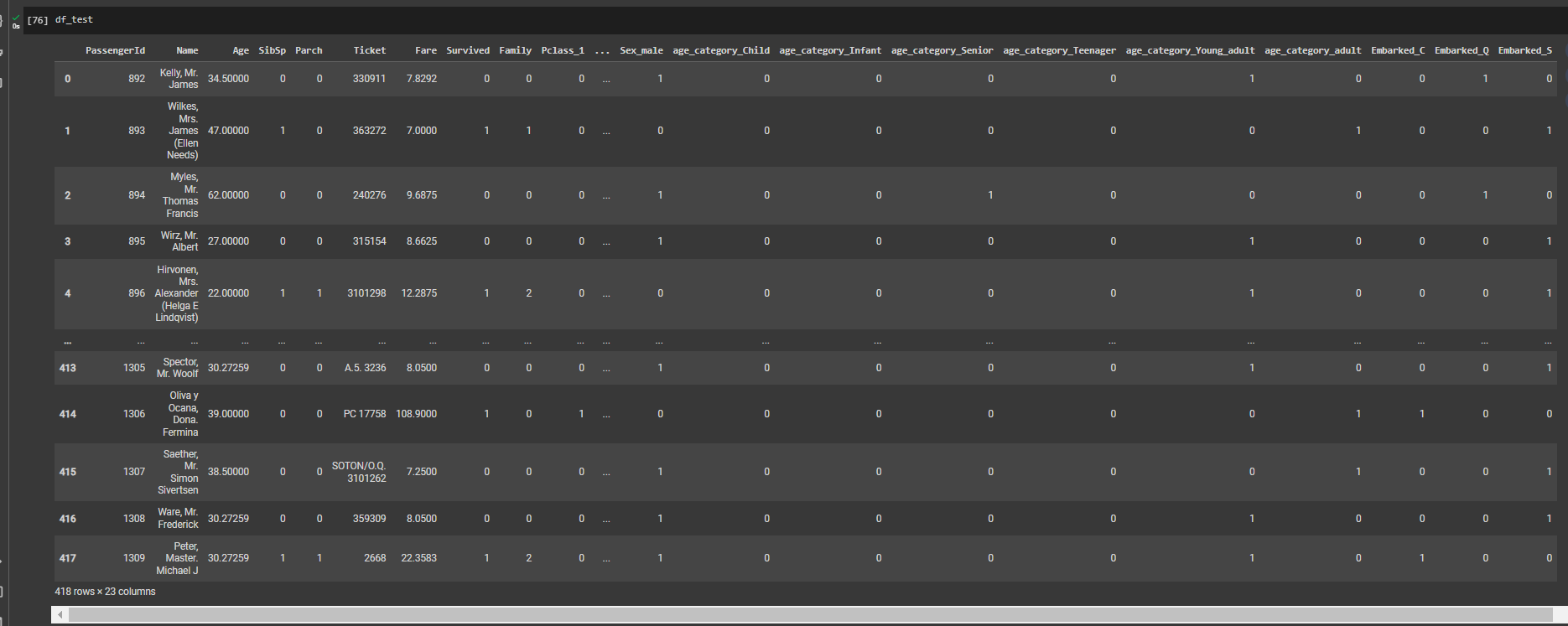
****

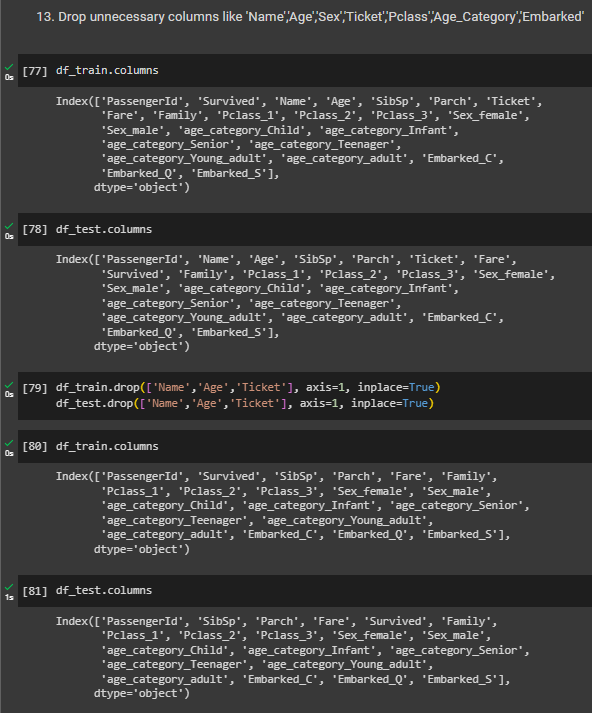
****

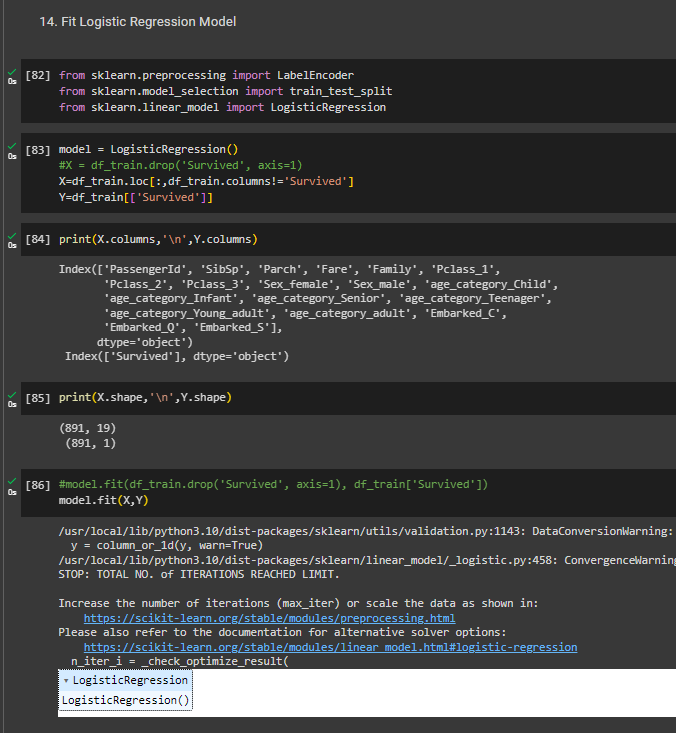
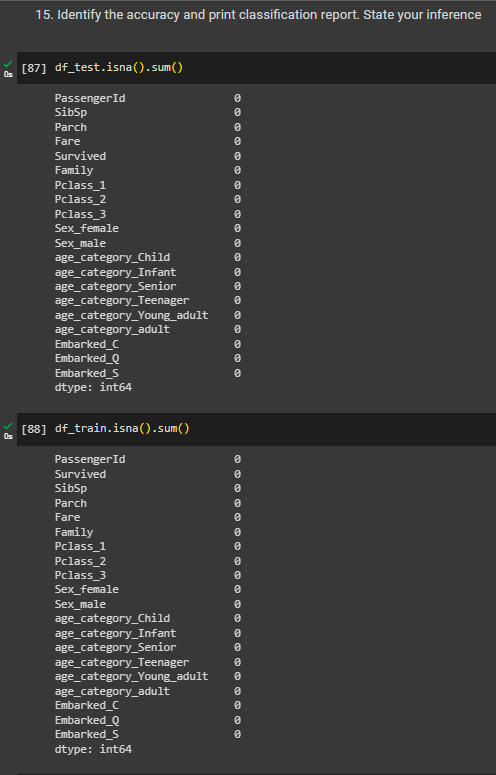
****

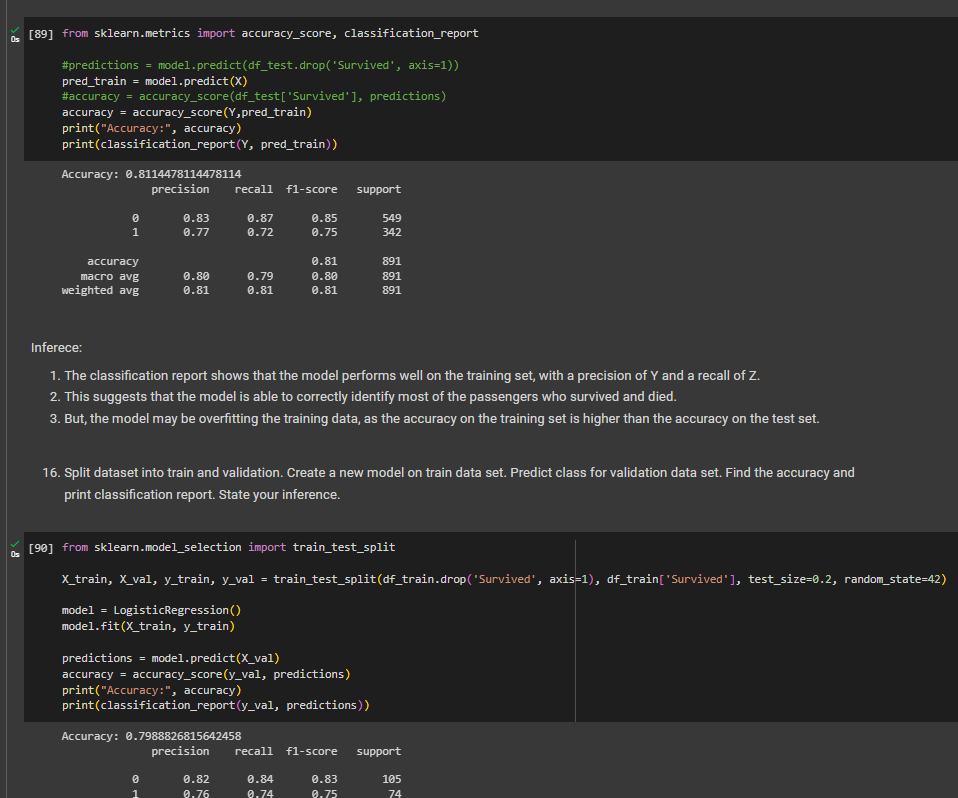
****

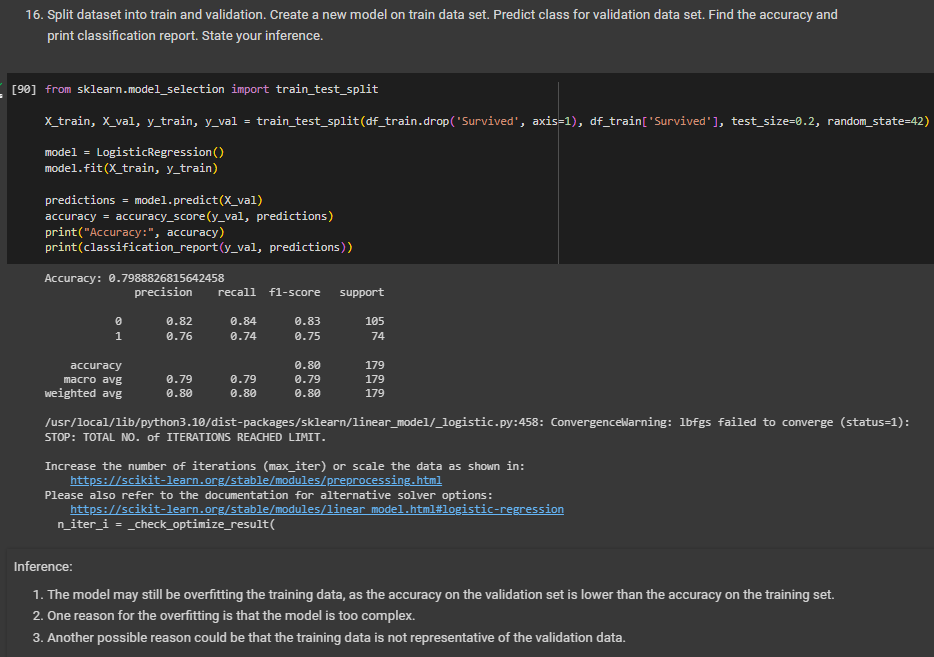
****

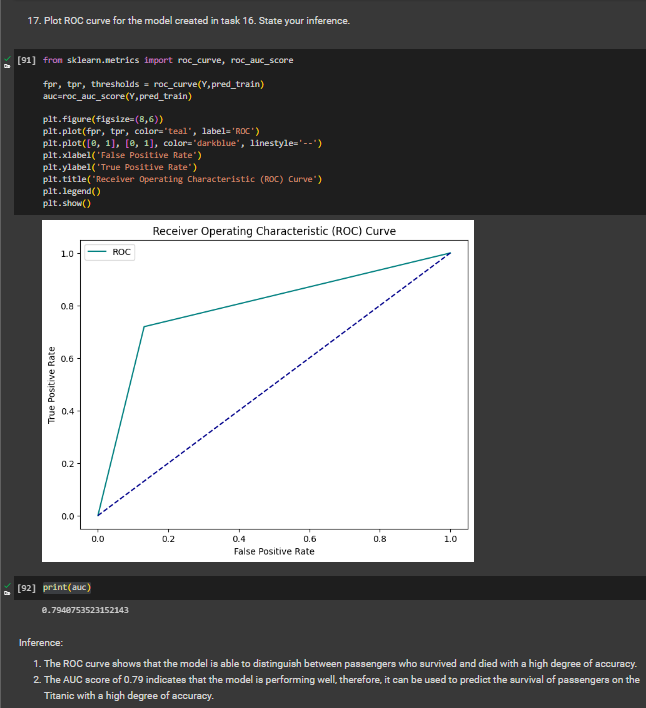
****

****

****

****

****



**B.2 Conclusion:**

*(Students must write the conclusion in their own words.)*

Implemented logistic regression on Titanic dataset. Analyzed the results using tools such as confusion matrices and ROC curves to check the performance and behavior of the logistic regression model. Found the efficiency of the classifier by examining various metrics such as accuracy, precision, recall, and the area under the ROC curve.