

SmartBridge Applied Data Science

Name: Routhu Sai Praneeth

Mail: routhusai.praneeth2020@vitstudent.ac.in

ADS Assignment 2

Titanic Ship Case Study:

Problem Description:

On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. Translated 32% survival rate.

- One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew.
- Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

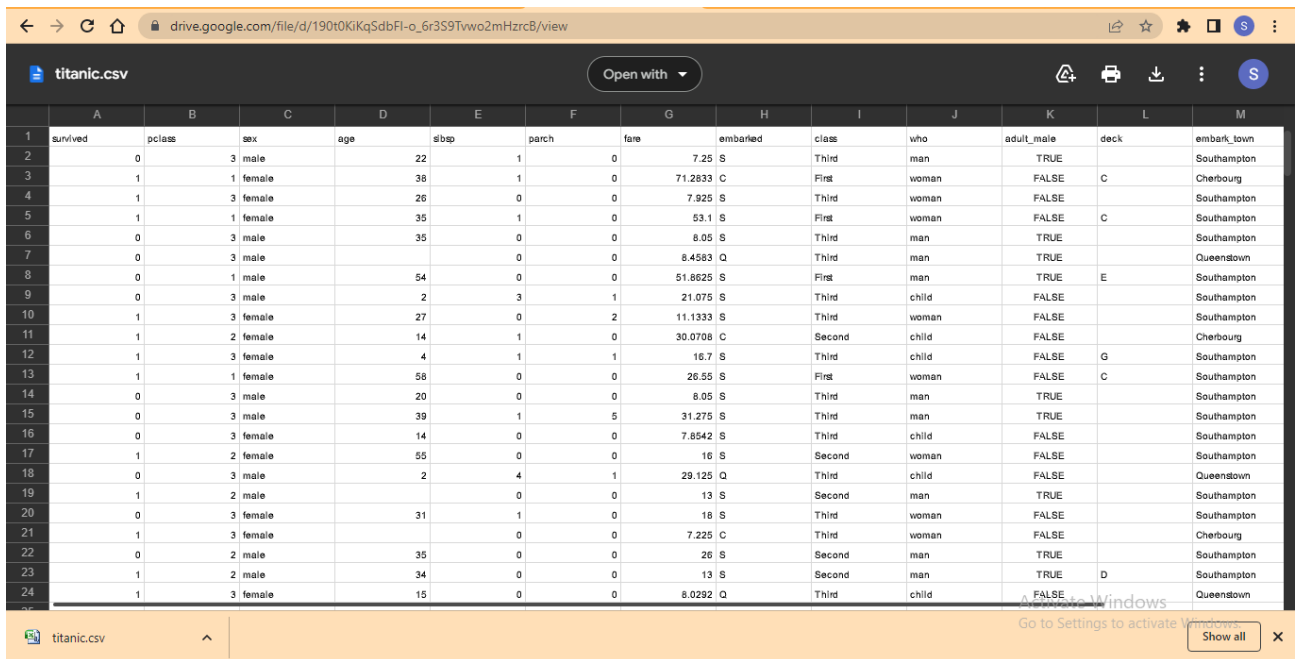
The problem associated with the Titanic dataset is to predict whether a passenger survived the disaster or not. The dataset contains various features such as passenger class, age, gender, cabin, fare, and whether the passenger had any siblings or spouses on board. These features can be used to build a predictive model to determine the likelihood of a passenger surviving the disaster. The dataset offers opportunities for feature engineering, data visualization, and model selection, making it a valuable resource for developing and testing data analysis and machine learning skills.

Drive Link to Colab File:

[Link](#)

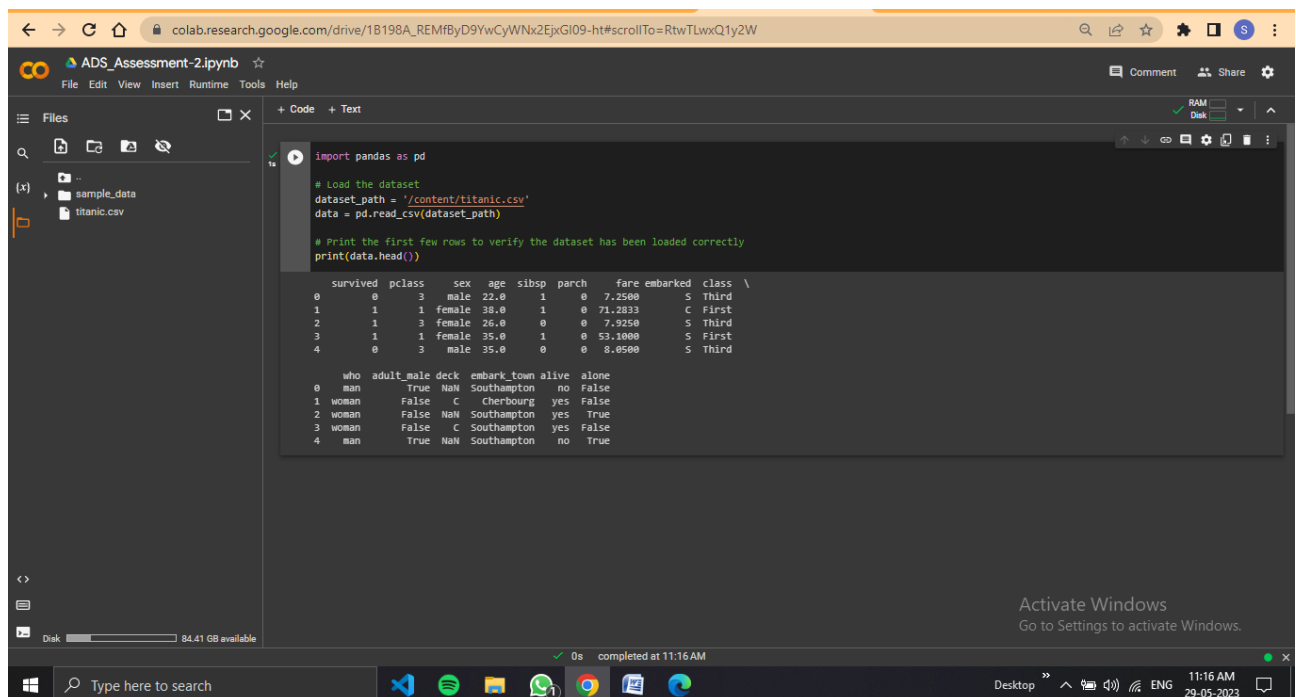
Tasks:

1. Downloading the dataset:



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town
2	0	3	male	22	1	0	7.25	S	Third	man	TRUE		Southampton
3	1	1	female	38	1	0	71.2833	C	First	woman	FALSE	C	Cherbourg
4	1	3	female	26	0	0	7.925	S	Third	woman	FALSE		Southampton
5	1	1	female	35	1	0	53.1	S	First	woman	FALSE	C	Southampton
6	0	3	male	35	0	0	8.05	S	Third	man	TRUE		Southampton
7	0	3	male		0	0	8.4583	Q	Third	man	TRUE		Queenstown
8	0	1	male	54	0	0	51.8625	S	First	man	TRUE	E	Southampton
9	0	3	male	2	3	1	21.075	S	Third	child	FALSE		Southampton
10	1	3	female	27	0	2	11.1333	S	Third	woman	FALSE		Southampton
11	1	2	female	14	1	0	30.0708	C	Second	child	FALSE		Cherbourg
12	1	3	female	4	1	1	16.7	S	Third	child	FALSE	G	Southampton
13	1	1	female	58	0	0	26.55	S	First	woman	FALSE	C	Southampton
14	0	3	male	20	0	0	8.05	S	Third	man	TRUE		Southampton
15	0	3	male	39	1	5	31.275	S	Third	man	TRUE		Southampton
16	0	3	female	14	0	0	7.8542	S	Third	child	FALSE		Southampton
17	1	2	female	55	0	0	16	S	Second	woman	FALSE		Southampton
18	0	3	male	2	4	1	29.125	Q	Third	child	FALSE		Queenstown
19	1	2	male		0	0	13	S	Second	man	TRUE		Southampton
20	0	3	female	31	1	0	16	S	Third	woman	FALSE		Southampton
21	1	3	female		0	0	7.225	C	Third	woman	FALSE		Cherbourg
22	0	2	male	35	0	0	26	S	Second	man	TRUE		Southampton
23	1	2	male	34	0	0	13	S	Second	man	TRUE	D	Southampton
24	1	3	female	15	0	0	8.0292	Q	Third	child	FALSE		Queenstown

2. Load the dataset.



```
import pandas as pd

# Load the dataset
dataset_path = '/content/titanic.csv'
data = pd.read_csv(dataset_path)

# Print the first few rows to verify the dataset has been loaded correctly
print(data.head())
```

survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone	
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

3.Perform Below Visualizations.

colab.research.google.com/drive/1B198A_REMfByD9YwCyWNx2EjxGI09-ht#scrollTo=eaDhCXvKDa0g

ADS_Assessment-2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

- sample_data
- titanic.csv

Visualization

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype  
---  -
 0   survived           891 non-null    int64  
 1   pclass             891 non-null    int64  
 2   sex                 891 non-null    object  
 3   age                 714 non-null    float64 
 4   sibsp              891 non-null    int64  
 5   parch              891 non-null    int64  
 6   fare               891 non-null    float64 
 7   embarked           889 non-null    object  
 8   class              891 non-null    object  
 9   who                 891 non-null    object  
10  adult_male         891 non-null    bool    
11  deck               203 non-null    object  
12  embark_town        889 non-null    object  
13  alive              891 non-null    object  
14  alone              891 non-null    bool    
dtypes: bool(2), float64(2), int64(4), object(7)
memory usage: 92.4+ KB
```

Activate Windows
Go to Settings to activate Windows.

0s completed at 11:19 AM

Desktop 29-05-2023 11:19 AM

• Univariate Analysis

colab.research.google.com/drive/1B198A_REMfByD9YwCyWNx2EjxGI09-ht#scrollTo=JifkcuTfEEw7

ADS_Assessment-2.ipynb

File Edit View Insert Runtime Tools Help

Comment Share

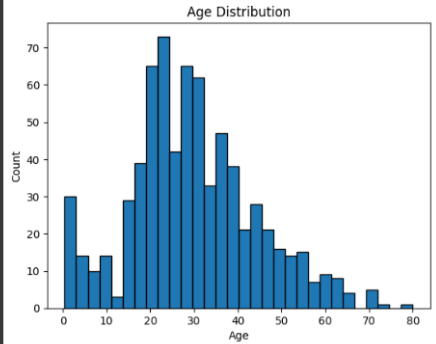
Files

- sample_data
- titanic.csv

Univariate Analysis

```
import matplotlib.pyplot as plt

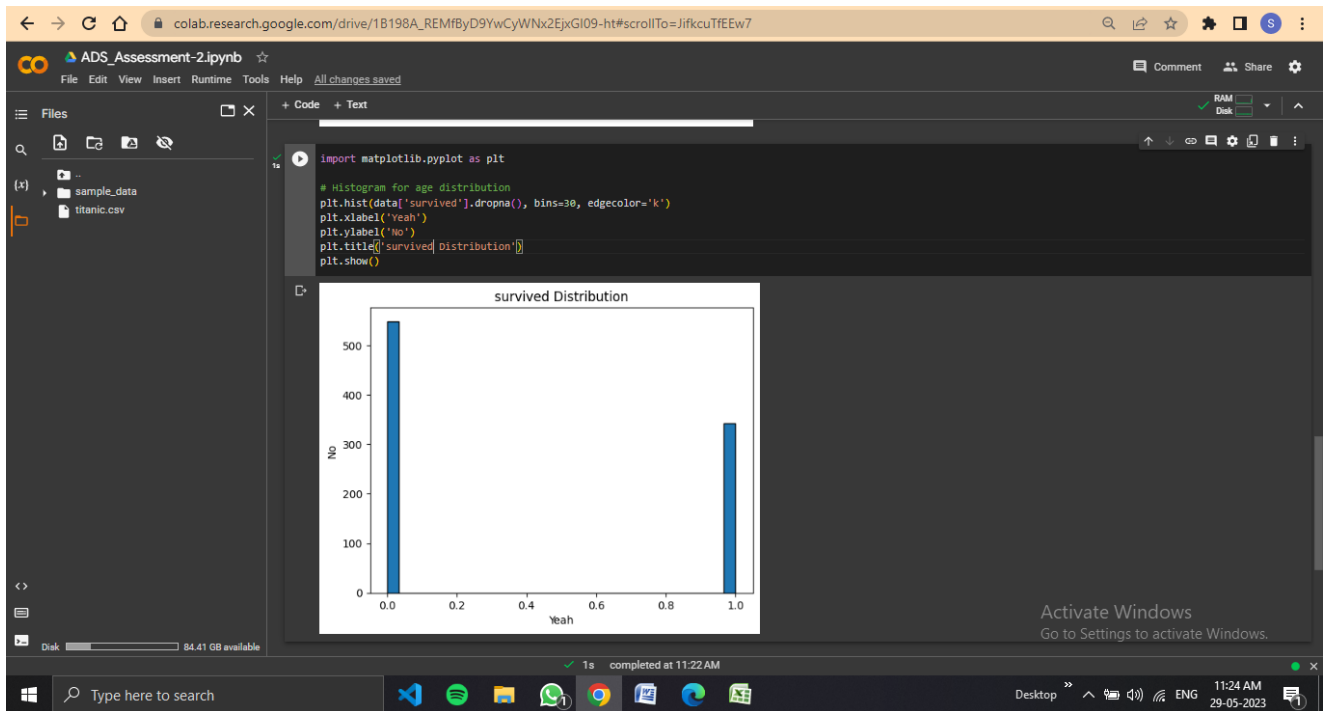
# Histogram for age distribution
plt.hist(data['age'].dropna(), bins=30, edgecolor='k')
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Age Distribution')
plt.show()
```



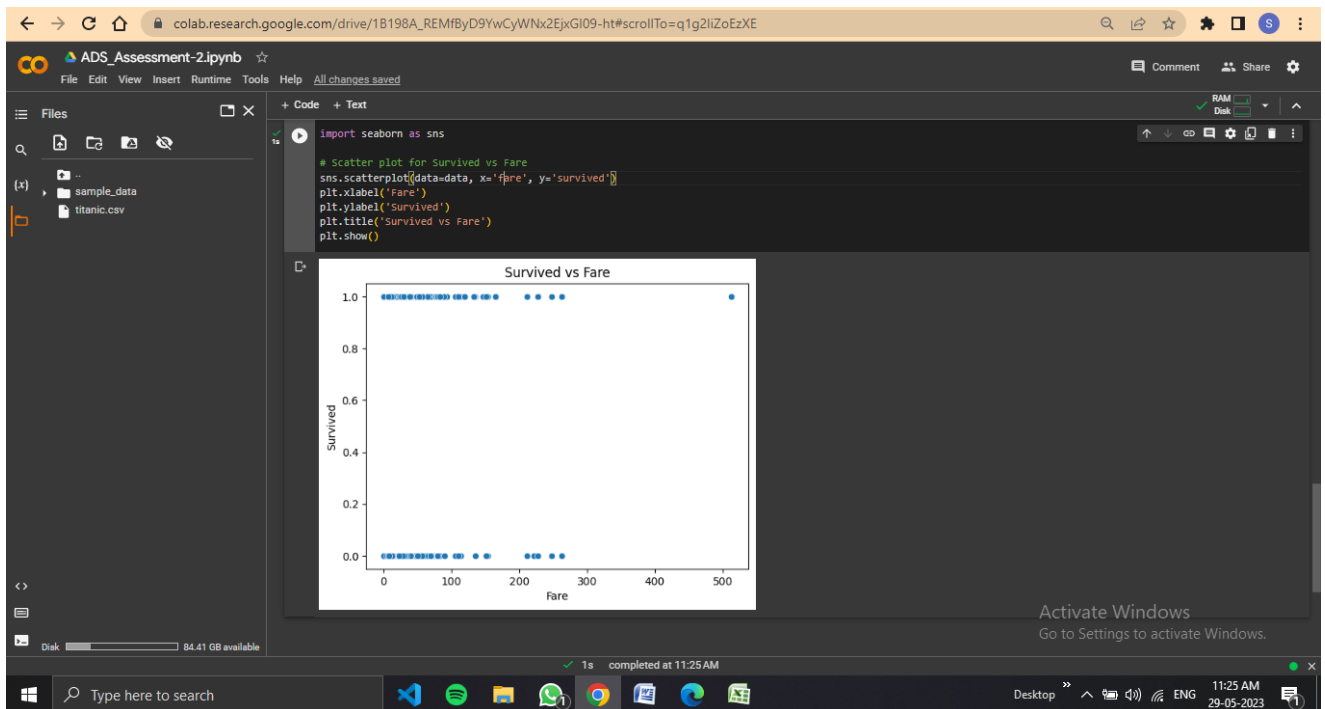
Activate Windows
Go to Settings to activate Windows.

1s completed at 11:22 AM

Desktop 29-05-2023 11:23 AM



• Bi - Variate Analysis



• Multi - Variate Analysis



4.Perform descriptive statistics on the dataset.

```
print(data.describe())
```

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

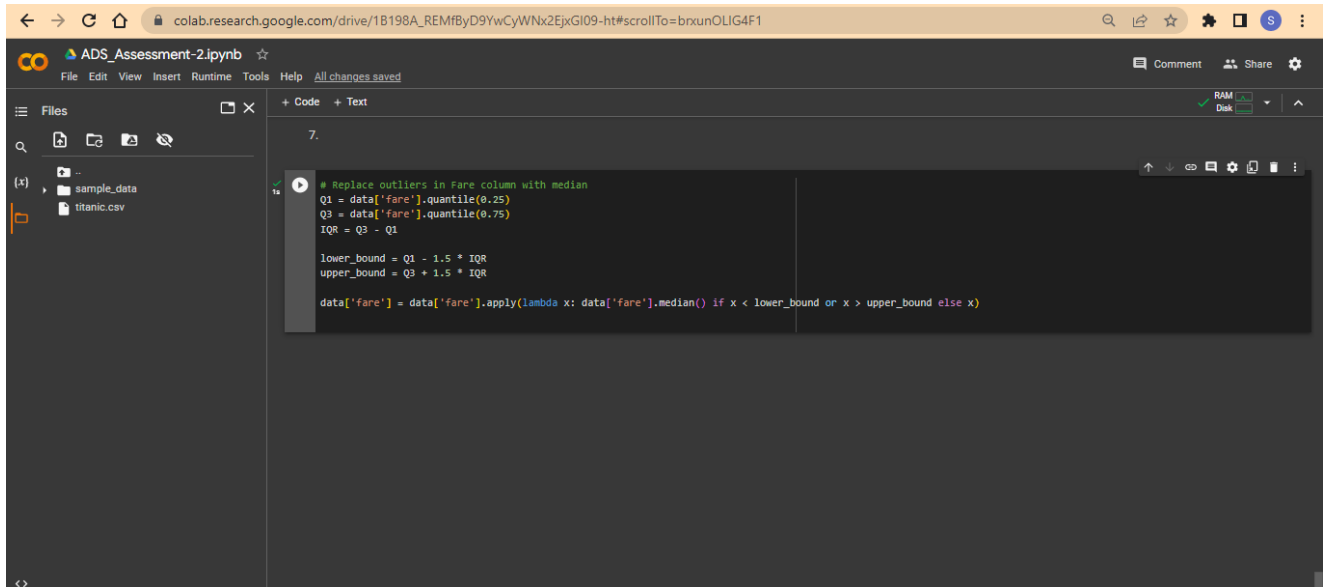
5.Handle the Missing values.

```
# Fill missing values in Age column with median
data['age'].fillna(data['age'].median(), inplace=True)

# Fill missing values in sibsp column with median
data['sibsp'].fillna(data['sibsp'].median(), inplace=True)

# Fill missing values in Age column with median
data['fare'].fillna(data['fare'].median(), inplace=True)
```

6. Find the outliers and replace the outliers



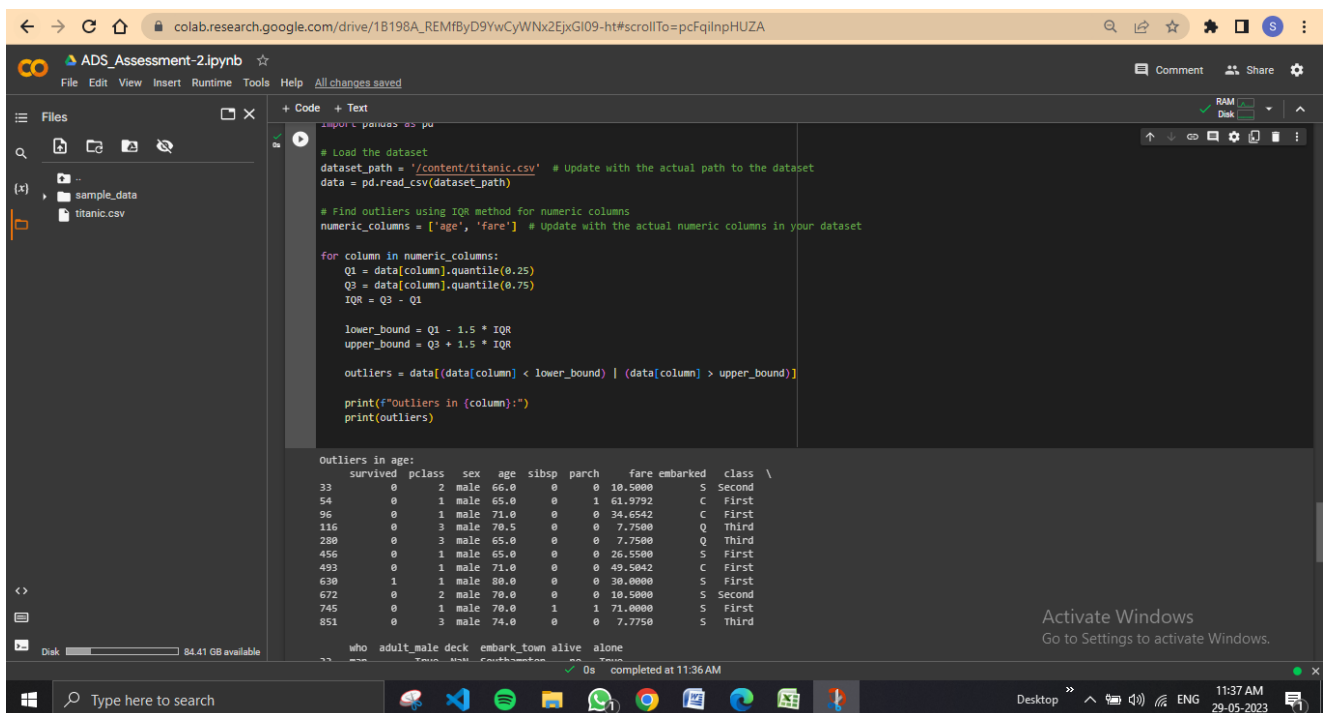
ADS_Assessment-2.ipynb

```
7.

# Replace outliers in fare column with median
Q1 = data['fare'].quantile(0.25)
Q3 = data['fare'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

data['fare'] = data['fare'].apply(lambda x: data['fare'].median() if x < lower_bound or x > upper_bound else x)
```



ADS_Assessment-2.ipynb

```
# Load the dataset
dataset_path = '/content/titanic.csv' # Update with the actual path to the dataset
data = pd.read_csv(dataset_path)

# Find outliers using IQR method for numeric columns
numeric_columns = ['age', 'fare'] # Update with the actual numeric columns in your dataset

for column in numeric_columns:
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = data[(data[column] < lower_bound) | (data[column] > upper_bound)]

    print(f"Outliers in {column}:")
    print(outliers)
```


Outliers in age:

survived	pclass	sex	age	sibsp	parch	fare	embarked	class
33	0	2	male	66.0	0	10.5000	S	Second
54	0	1	male	65.0	1	61.9792	C	First
96	0	1	male	71.0	0	34.6542	C	First
116	0	3	male	70.5	0	7.7500	Q	Third
280	0	3	male	65.0	0	7.7500	Q	Third
456	0	1	male	65.0	0	26.5500	S	First
493	0	1	male	71.0	0	49.5042	C	First
630	1	1	male	80.0	0	30.0000	S	First
672	0	2	male	70.0	0	10.5000	S	Second
745	0	1	male	70.0	1	71.0000	S	First
851	0	3	male	74.0	0	7.7750	S	Third

who adult_male deck embark_town alive alone

Activate Windows
Go to Settings to activate Windows.

completed at 11:36 AM



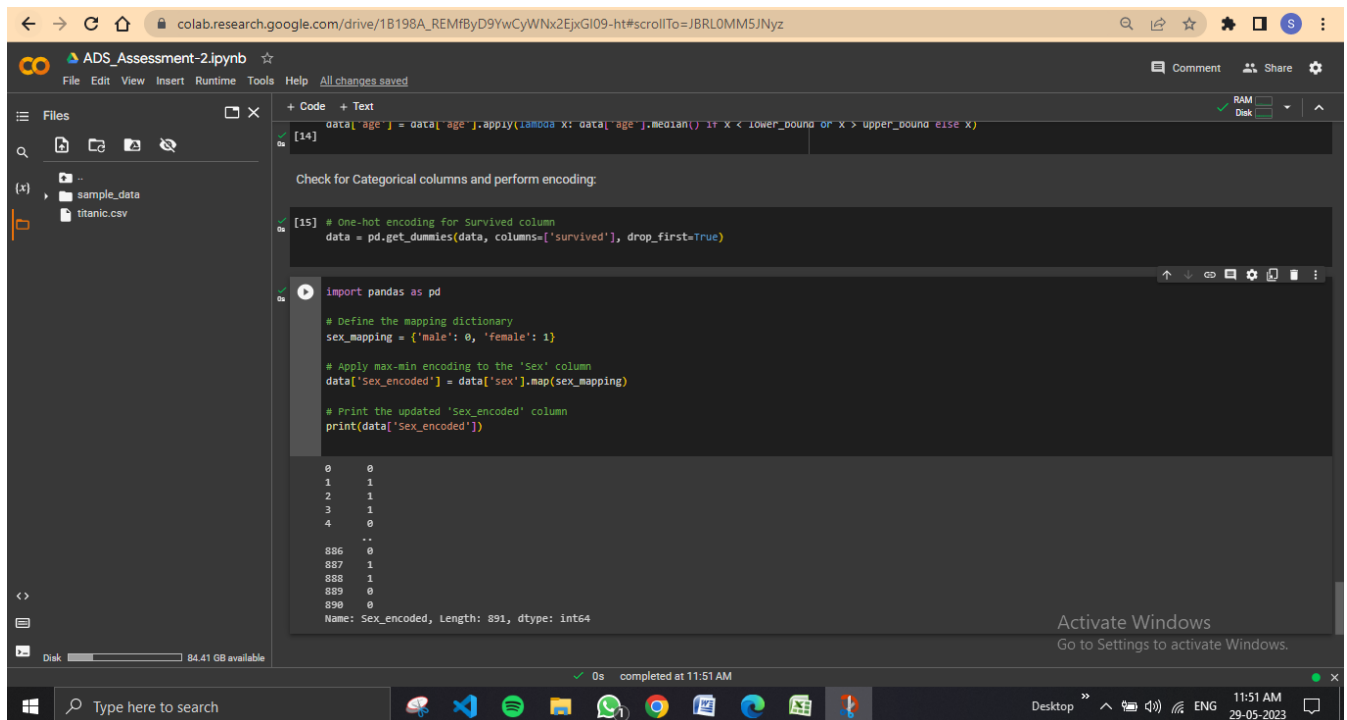
ADS_Assessment-2.ipynb

```
# Replace outliers in age column with median
Q1 = data['age'].quantile(0.25)
Q3 = data['age'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

data['age'] = data['age'].apply(lambda x: data['age'].median() if x < lower_bound or x > upper_bound else x)
```

7. Check for Categorical columns and perform encoding.



```
data["age"] = data["age"].apply(lambda x: data["age"].median() if x < lower_bound or x > upper_bound else x)
```

Check for Categorical columns and perform encoding:

```
[15] # One-hot encoding for Survived column
data = pd.get_dummies(data, columns=['survived'], drop_first=True)
```

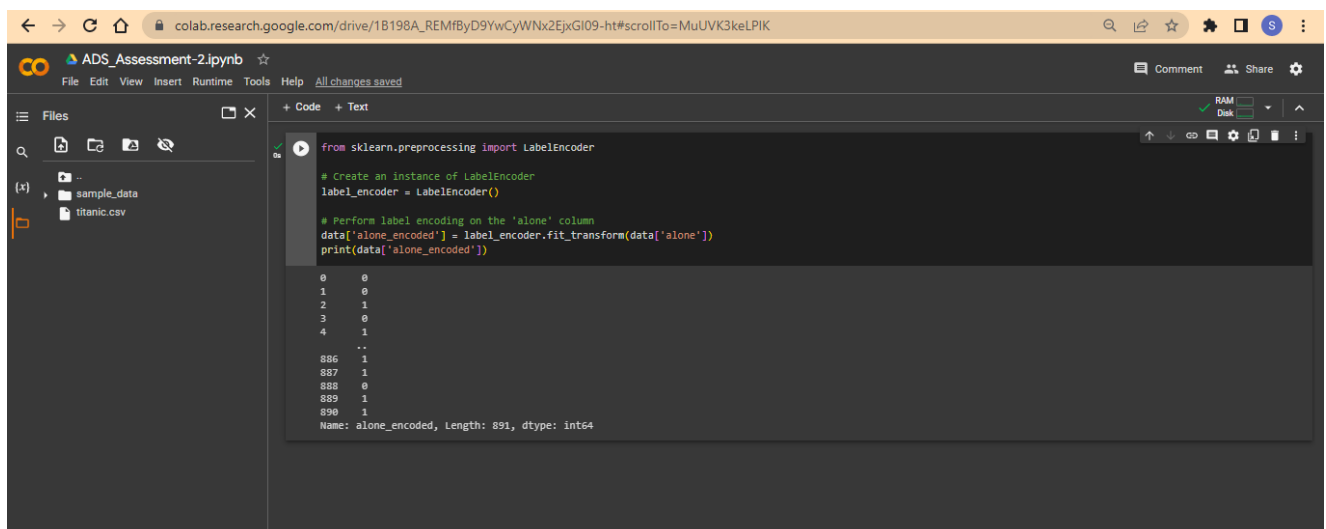
```
import pandas as pd

# Define the mapping dictionary
sex_mapping = {'male': 0, 'female': 1}

# Apply max-min encoding to the 'Sex' column
data["Sex_encoded"] = data["Sex"].map(sex_mapping)

# Print the updated 'Sex_encoded' column
print(data["Sex_encoded"])
```

```
0      0
1      1
2      1
3      1
4      0
..
886     0
887     1
888     1
889     0
890     0
Name: Sex_encoded, Length: 891, dtype: int64
```



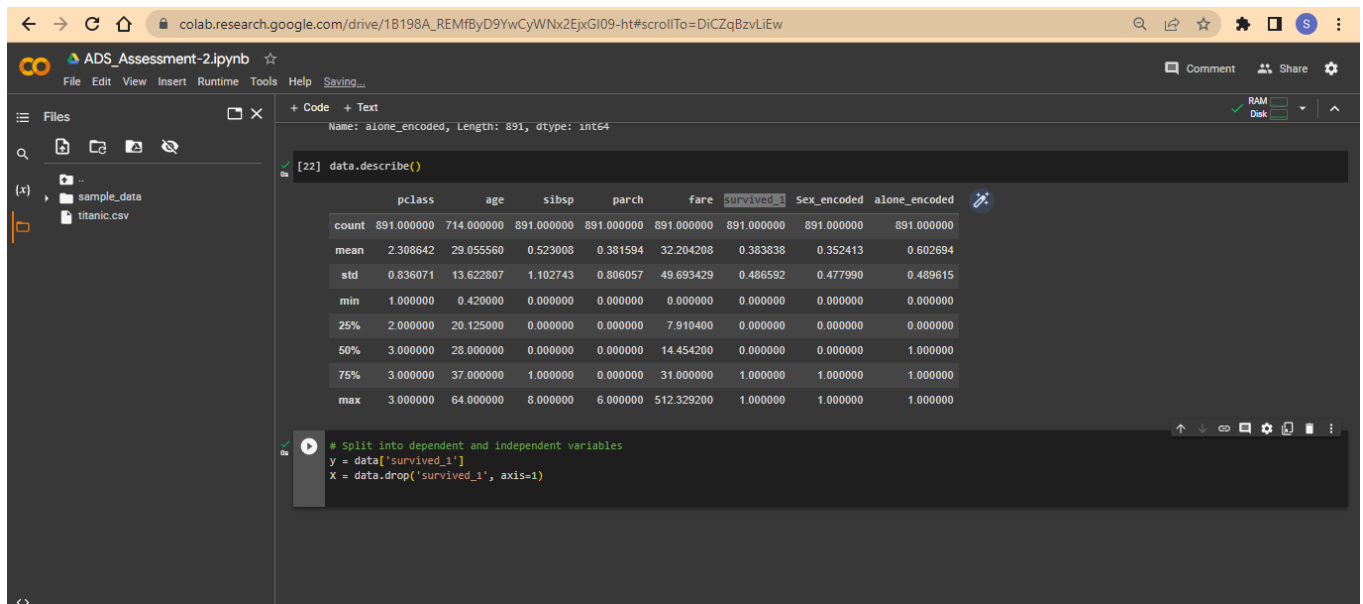
```
from sklearn.preprocessing import LabelEncoder

# Create an instance of LabelEncoder
label_encoder = LabelEncoder()

# Perform label encoding on the 'alone' column
data["alone_encoded"] = label_encoder.fit_transform(data["alone"])
print(data["alone_encoded"])
```

```
0      0
1      0
2      1
3      0
4      1
..
886     1
887     1
888     0
889     1
890     1
Name: alone_encoded, Length: 891, dtype: int64
```

8. Split the data into dependent and independent variables.



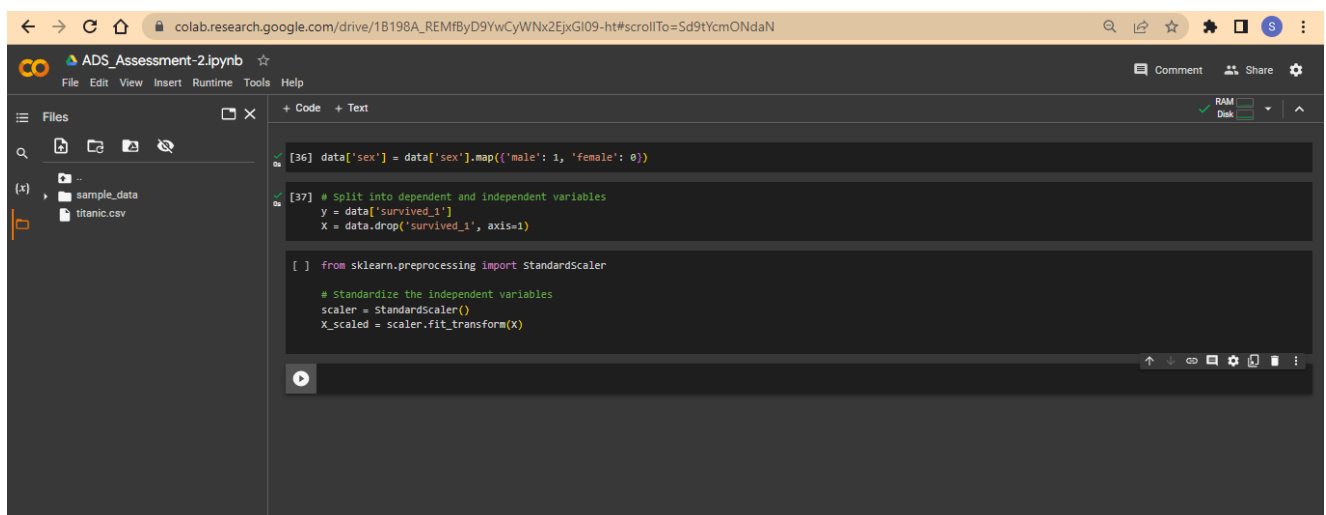
The screenshot shows a Google Colab notebook titled 'ADS_Assessment-2.ipynb'. The left sidebar shows a file explorer with 'sample_data' and 'titanic.csv'. The main code area has two cells. The first cell, labeled [22], contains `data.describe()` and displays a summary table for the 'survived_1' variable. The second cell contains code to split the data into dependent and independent variables.

	pclass	age	sibsp	parch	fare	survived_1	Sex_encoded	alone_encoded
count	891.000000	714.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	2.308642	29.055560	0.523008	0.381594	32.204208	0.383838	0.352413	0.602694
std	0.836071	13.622807	1.102743	0.806057	49.693429	0.486592	0.477990	0.489615
min	1.000000	0.420000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2.000000	20.125000	0.000000	0.000000	7.910400	0.000000	0.000000	0.000000
50%	3.000000	28.000000	0.000000	0.000000	14.454200	0.000000	0.000000	1.000000
75%	3.000000	37.000000	1.000000	0.000000	31.000000	1.000000	1.000000	1.000000
max	3.000000	64.000000	8.000000	6.000000	512.329200	1.000000	1.000000	1.000000

```
[22] data.describe()

# Split into dependent and independent variables
y = data['survived_1']
X = data.drop('survived_1', axis=1)
```

9. Scale the independent variables.



The screenshot shows the same Colab notebook with additional code to standardize the independent variables. The first cell, labeled [36], maps the 'sex' variable to 1 for 'male' and 0 for 'female'. The second cell, labeled [37], repeats the splitting of the data. The third cell imports `StandardScaler` from `sklearn.preprocessing` and applies it to the independent variables.

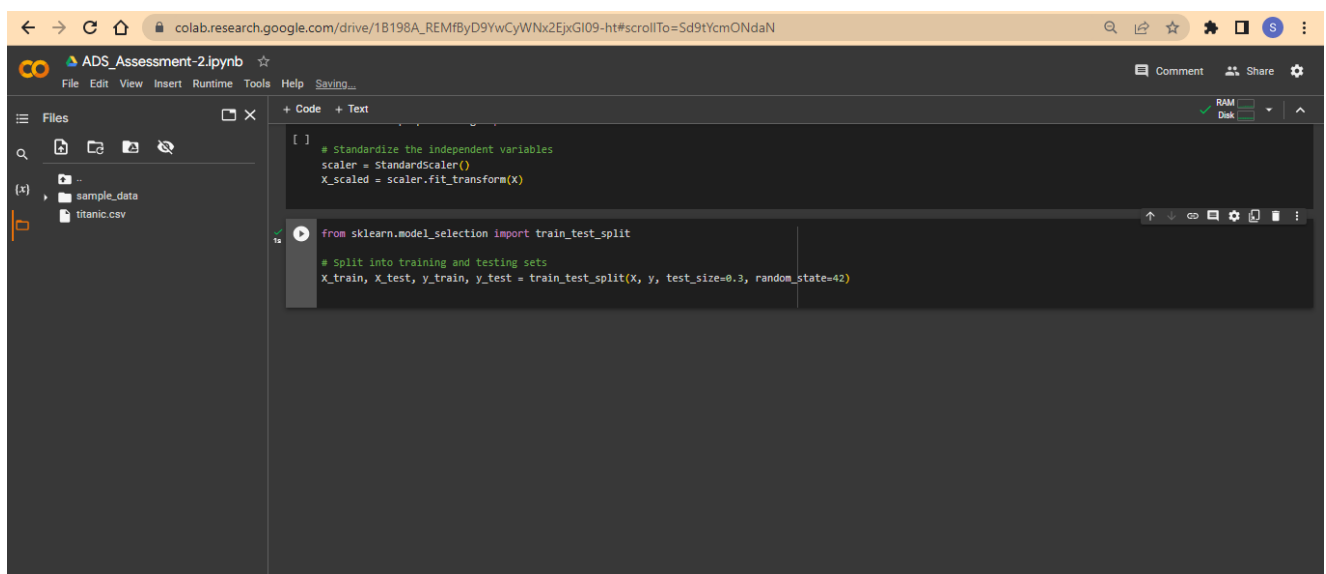
```
[36] data['sex'] = data['sex'].map({'male': 1, 'female': 0})

[37] # Split into dependent and independent variables
y = data['survived_1']
X = data.drop('survived_1', axis=1)

[ ] from sklearn.preprocessing import StandardScaler

# Standardize the independent variables
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

10. Split the data into training and testing



The screenshot shows the same Colab notebook with code to split the data into training and testing sets. The first cell, labeled [], repeats the standardization of the independent variables. The second cell, labeled [], imports `train_test_split` from `sklearn.model_selection` and splits the data into training and testing sets.

```
[ ] # Standardize the independent variables
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

[ ] from sklearn.model_selection import train_test_split

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```