

Task-4

Titanic Classification

Samritha A.R



Importing all necessary Libraries & Loading Datasets

The screenshot shows a Google Colab notebook titled "Titanic Classification.ipynb". The notebook interface includes a toolbar at the top with icons for file operations, a search bar, and user profile. The main workspace is organized into sections: "Task-4" and "Titanic Classification". The "Importing all required Libaries" section contains the following Python code:

```
[1] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from warnings import filterwarnings
filterwarnings('ignore')
```

The "Loading Datasets" section contains the following code:

```
[2] train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')

[3] pd.set_option('display.max_columns',10,'display.width',1000)
train.head()
```

A status message at the bottom indicates: "Connected to Python 3 Google Compute Engine backend".

Fetching 1st 5 rows of train & test datasets

Titanic Classification.ipynb

```
[3]: pd.set_option('display.max_columns',10,'display.width',1000)
train.head()
```

PassengerId	Survived	Pclass	Name	Sex	...	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	Braund, Mr. Owen Harris	male	...	0	A/5 21171	7.2500	Nan	S
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	...	0	PC 17599	71.2833	C85	C
2	3	1	3	female	...	0	STON/O2. 3101282	7.9250	Nan	S
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	...	0	113803	53.1000	C123	S
4	5	0	Allen, Mr. William Henry	male	...	0	373450	8.0500	Nan	S

5 rows × 12 columns

```
[4]: pd.set_option('display.max_columns',10,'display.width',1000)
test.head()
```

PassengerId	Pclass	Name	Sex	Age	...	Parch	Ticket	Fare	Cabin	Embarked
0	892	3 Kelly, Mr. James	male	34.5	...	0	330911	7.8292	Nan	Q
1	893	3 Wilkes, Mrs. James (Ellen Needs)	female	47.0	...	0	363272	7.0000	Nan	S
2	894	2 Myles, Mr. Thomas Francis	male	62.0	...	0	240276	9.6875	Nan	Q
3	895	3 Wirz, Mr. Albert	male	27.0	...	0	315154	8.6625	Nan	S
4	896	3 Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	...	1	3101298	12.2875	Nan	S

5 rows × 11 columns

Getting Train dataset Info & Shape

The screenshot shows a Jupyter Notebook interface with the title "Titanic Classification.ipynb". The notebook has a sidebar with various icons for file operations, search, and help. The main area contains two code cells:

```
[5] train.info()  
train.shape  
  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 891 entries, 0 to 890  
Data columns (total 12 columns):  
 #   Column      Non-Null Count  Dtype    
 ---  --          --          --  
 0   PassengerId 891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object  
 4   Sex          891 non-null    object  
 5   Age          714 non-null    float64  
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object  
 9   Fare          891 non-null    float64  
 10  Cabin         204 non-null    object  
 11  Embarked      889 non-null    object  
dtypes: float64(2), int64(5), object(5)  
memory usage: 83.7+ KB  
(891, 12)  
  
[6] test.info()  
test.shape  
  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 418 entries, 0 to 417
```

Getting Test dataset Info & Shape

The screenshot shows a Jupyter Notebook interface with the title "Titanic Classification.ipynb". The notebook contains the following code cells:

- Cell [5]:

```
0s      Embarked    object
[5] dtypes: float64(2), int64(5), object(5)
      memory usage: 83.7+ KB
      (891, 12)
```
- Cell [6]:

```
0s
[6] test.info()
test.shape

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   PassengerId  418 non-null    int64  
 1   Pclass        418 non-null    int64  
 2   Name          418 non-null    object  
 3   Sex           418 non-null    object  
 4   Age           332 non-null    float64 
 5   SibSp         418 non-null    int64  
 6   Parch         418 non-null    int64  
 7   Ticket        418 non-null    object  
 8   Fare          417 non-null    float64 
 9   Cabin         91 non-null    object  
 10  Embarked      418 non-null    object  
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
(418, 11)
```
- Cell [7]:

```
0s
[7] train.isnull().sum()

PassengerId      0
Survived         0
```

The notebook interface includes a sidebar with file management icons, a toolbar with various tools, and a status bar at the top right showing RAM and Disk usage.

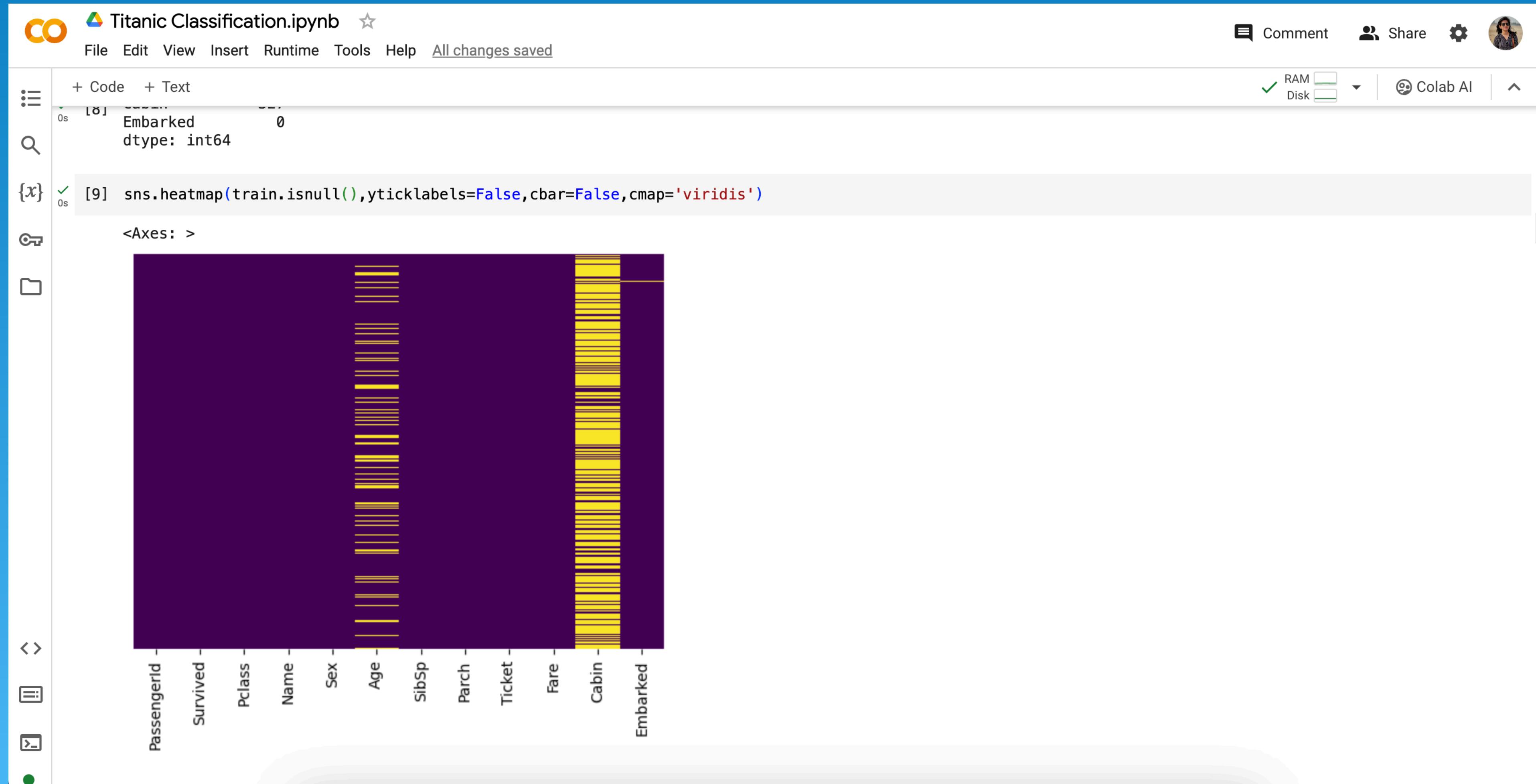
Checking for Null values

The screenshot shows a Jupyter Notebook interface with the title "Titanic Classification.ipynb". The notebook has a sidebar on the left containing icons for file operations, search, and cell types. The main area displays two code cells. The first cell, labeled [7], contains the command `train.isnull().sum()` and its output, which is a Series showing the count of null values for each column in the training dataset. The second cell, labeled [8], contains the command `test.isnull().sum()` and its output, which is a Series showing the count of null values for each column in the testing dataset. Both outputs show zero null values across all columns.

```
[7] train.isnull().sum()
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked        2
dtype: int64

[8] test.isnull().sum()
PassengerId      0
Pclass           0
Name             0
Sex              0
Age            86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked        0
dtype: int64
```

Heatmap for Train set to check null values



Description of Train set

CO Titanic Classification.ipynb ★

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙️ 

+ Code + Text RAM Disk Colab AI ▾

☰ Description of Datasets {x}

[10] train.describe(include="all")

	PassengerId	Survived	Pclass	Name	Sex	...	Parch	Ticket	Fare	Cabin	Embarked
count	891.000000	891.000000	891.000000	891	891	...	891.000000	891	891.000000	204	889
unique	Nan	Nan	Nan	891	2	...	Nan	681	Nan	147	3
top	Nan	Nan	Nan	Braund, Mr. Owen Harris	male	...	Nan	347082	Nan	B96 B98	S
freq	Nan	Nan	Nan	1	577	...	Nan	7	Nan	4	644
mean	446.000000	0.383838	2.308642	Nan	Nan	...	0.381594	Nan	32.204208	Nan	Nan
std	257.353842	0.486592	0.836071	Nan	Nan	...	0.806057	Nan	49.693429	Nan	Nan
min	1.000000	0.000000	1.000000	Nan	Nan	...	0.000000	Nan	0.000000	Nan	Nan
25%	223.500000	0.000000	2.000000	Nan	Nan	...	0.000000	Nan	7.910400	Nan	Nan
50%	446.000000	0.000000	3.000000	Nan	Nan	...	0.000000	Nan	14.454200	Nan	Nan
75%	668.500000	1.000000	3.000000	Nan	Nan	...	0.000000	Nan	31.000000	Nan	Nan
max	891.000000	1.000000	3.000000	Nan	Nan	...	6.000000	Nan	512.329200	Nan	Nan

11 rows × 12 columns

[11] test.describe(include="all")

	PassengerId	Pclass	Name	Sex	Age	...	Parch	Ticket	Fare	Cabin	Embarked
--	-------------	--------	------	-----	-----	-----	-------	--------	------	-------	----------

Description of Train set

Titanic Classification.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

RAM Disk Colab AI

[11] test.describe(include="all")

	PassengerId	Pclass	Name	Sex	Age	...	Parch	Ticket	Fare	Cabin	Embarked
count	418.000000	418.000000	418	418	332.000000	...	418.000000	418	417.000000	91	418
unique	Nan	Nan	418	2	Nan	...	Nan	363	Nan	76	3
top	Nan	Nan	Kelly, Mr. James	male	Nan	...	Nan	PC 17608	Nan	B57 B59 B63 B66	S
freq	Nan	Nan	1	266	Nan	...	Nan	5	Nan	3	270
mean	1100.500000	2.265550	Nan	Nan	30.272590	...	0.392344	Nan	35.627188	Nan	Nan
std	120.810458	0.841838	Nan	Nan	14.181209	...	0.981429	Nan	55.907576	Nan	Nan
min	892.000000	1.000000	Nan	Nan	0.170000	...	0.000000	Nan	0.000000	Nan	Nan
25%	996.250000	1.000000	Nan	Nan	21.000000	...	0.000000	Nan	7.895800	Nan	Nan
50%	1100.500000	3.000000	Nan	Nan	27.000000	...	0.000000	Nan	14.454200	Nan	Nan
75%	1204.750000	3.000000	Nan	Nan	39.000000	...	0.000000	Nan	31.500000	Nan	Nan
max	1309.000000	3.000000	Nan	Nan	76.000000	...	9.000000	Nan	512.329200	Nan	Nan

11 rows x 11 columns

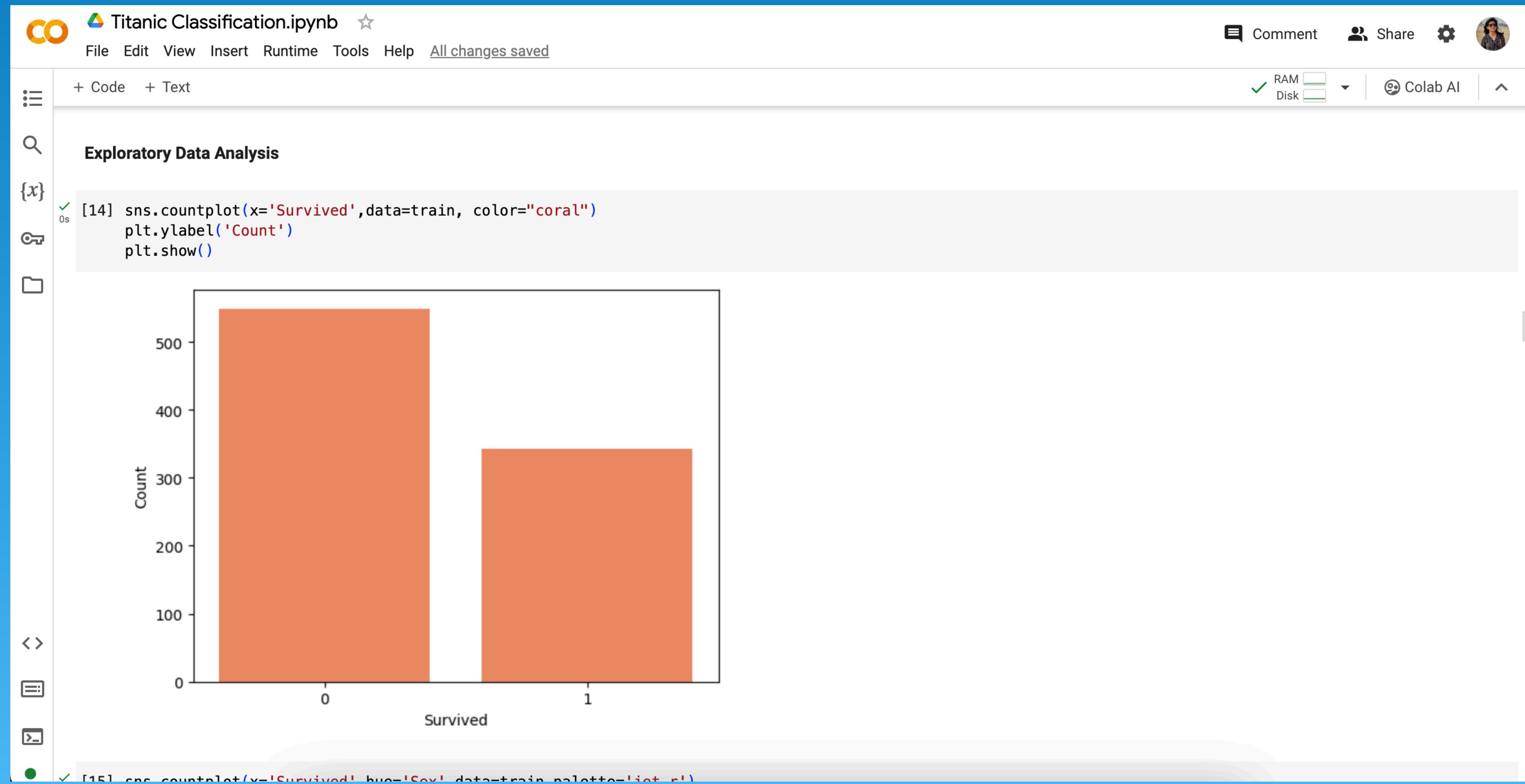
[12] train.groupby('Survived').mean()

	PassengerId	Pclass	Age	SibSp	Parch	Fare
Survived						
0	447.016393	2.531876	30.626179	0.553734	0.329690	22.117887
1	444.368421	1.950292	28.343690	0.473684	0.464912	48.395408

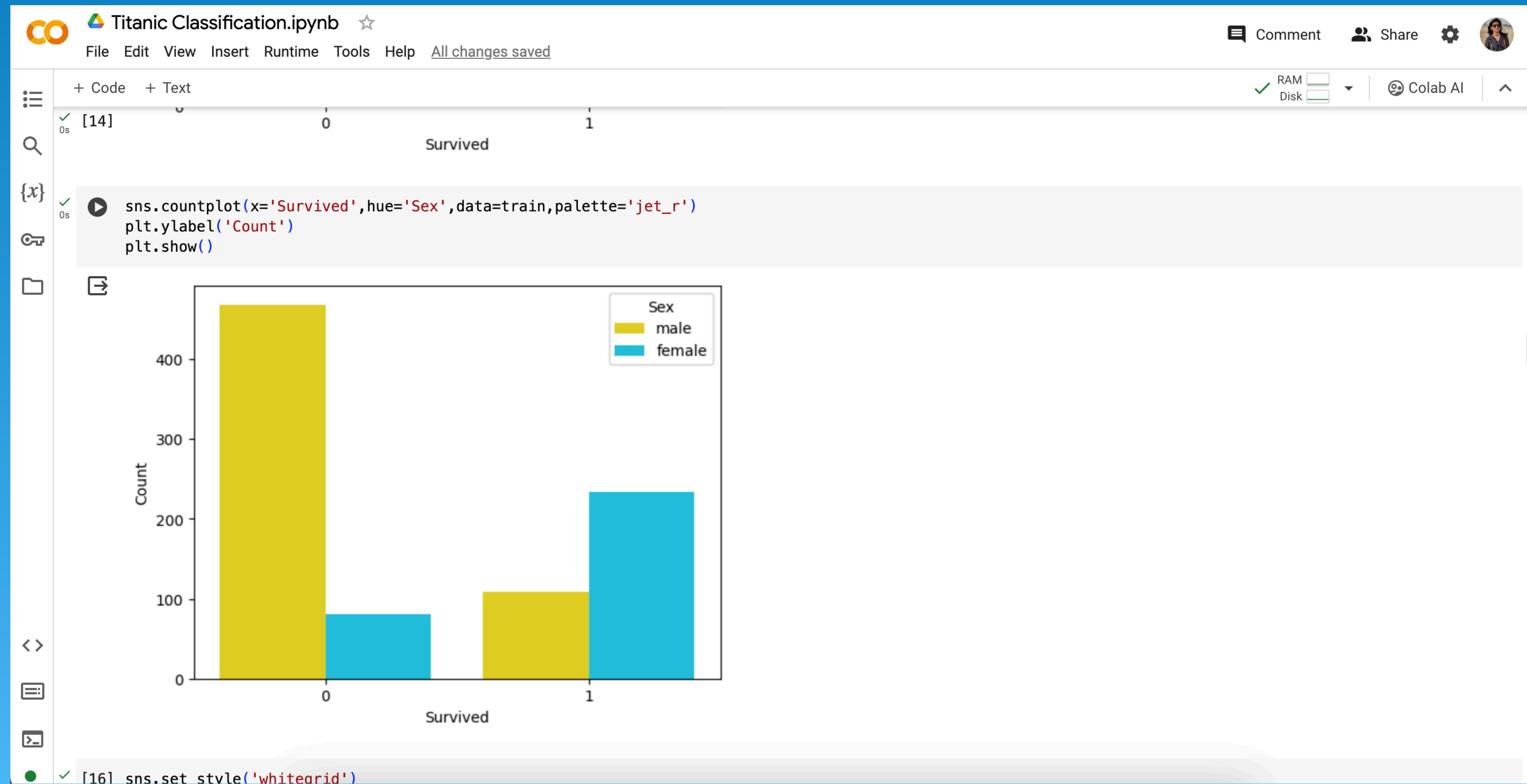
Correlation Plot of Train dataset



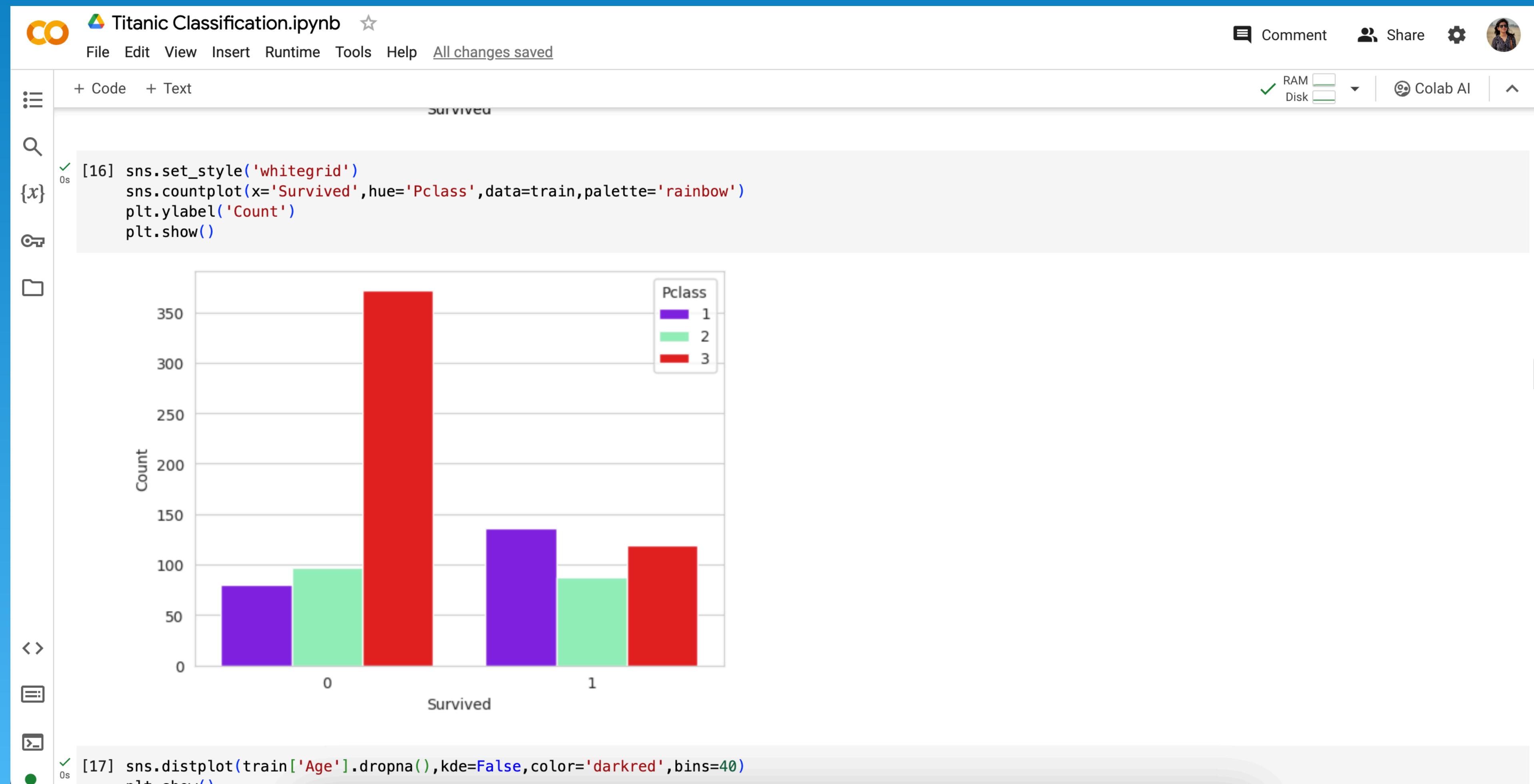
Exploratory Data Analysis for Train data



Exploratory Data Analysis for Train data



Exploratory Data Analysis for Train data



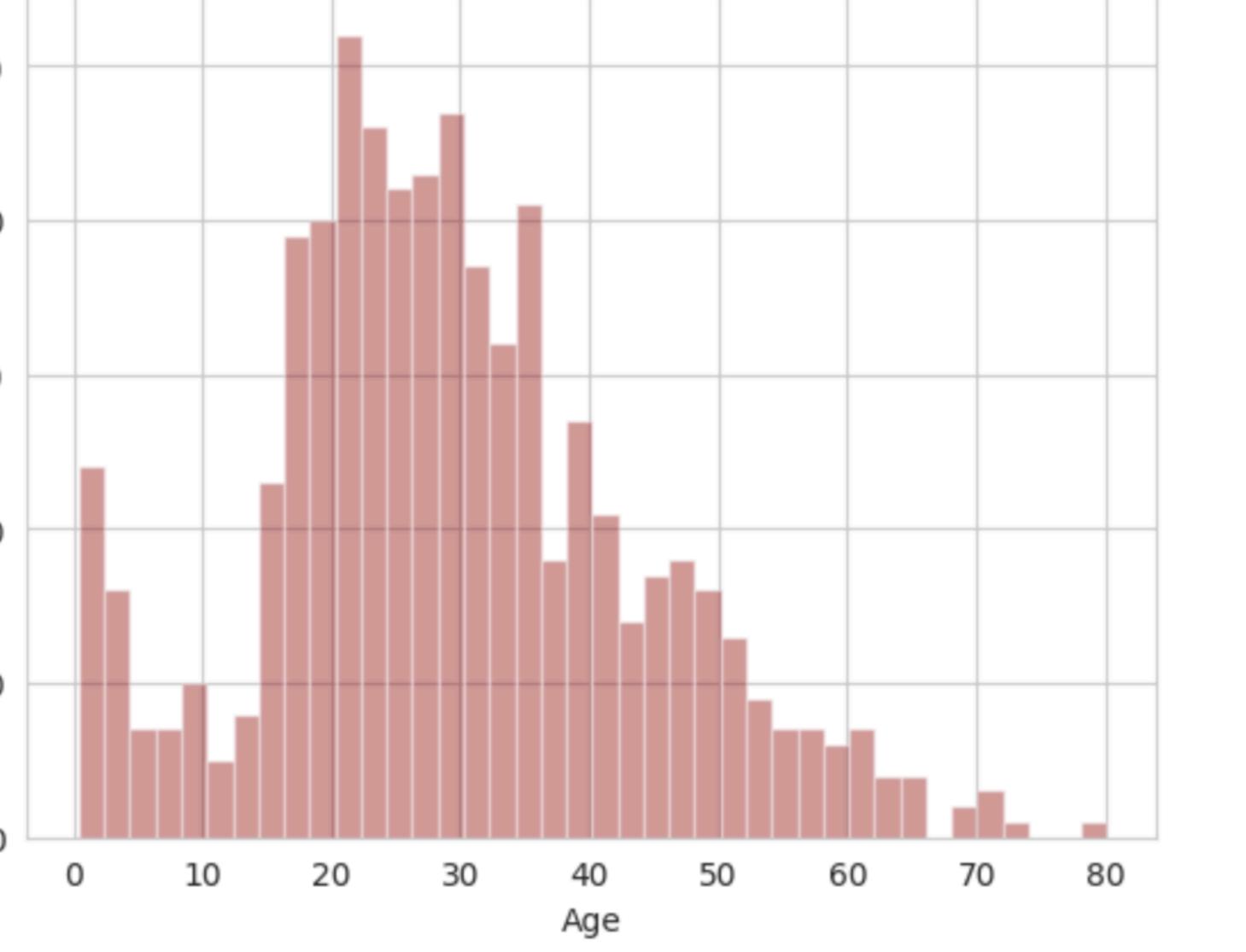
Exploratory Data Analysis for Train data

 Titanic Classification.ipynb 

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text RAM Disk Colab AI

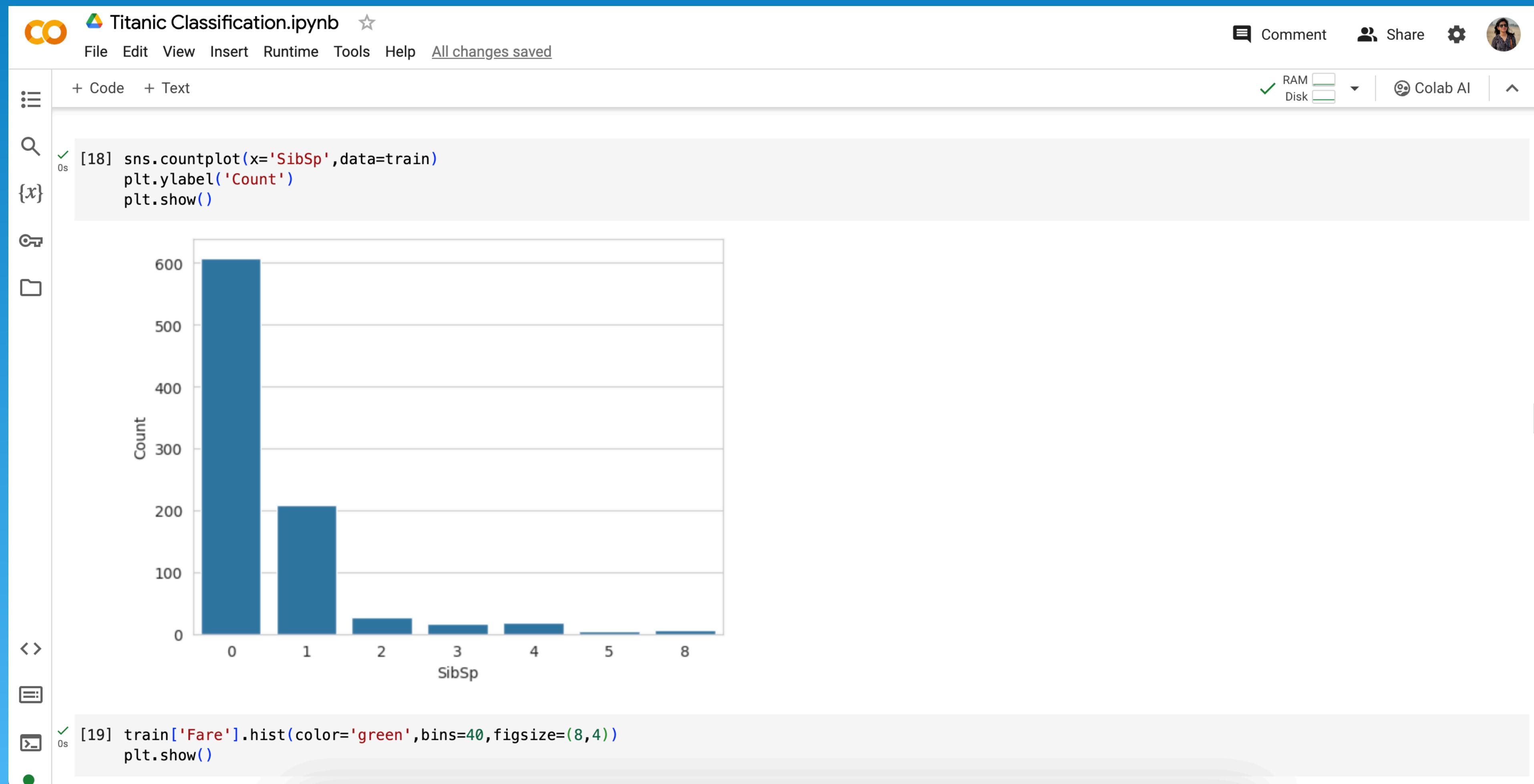
[17] sns.distplot(train['Age'].dropna(), kde=False, color='darkred', bins=40)
plt.show()



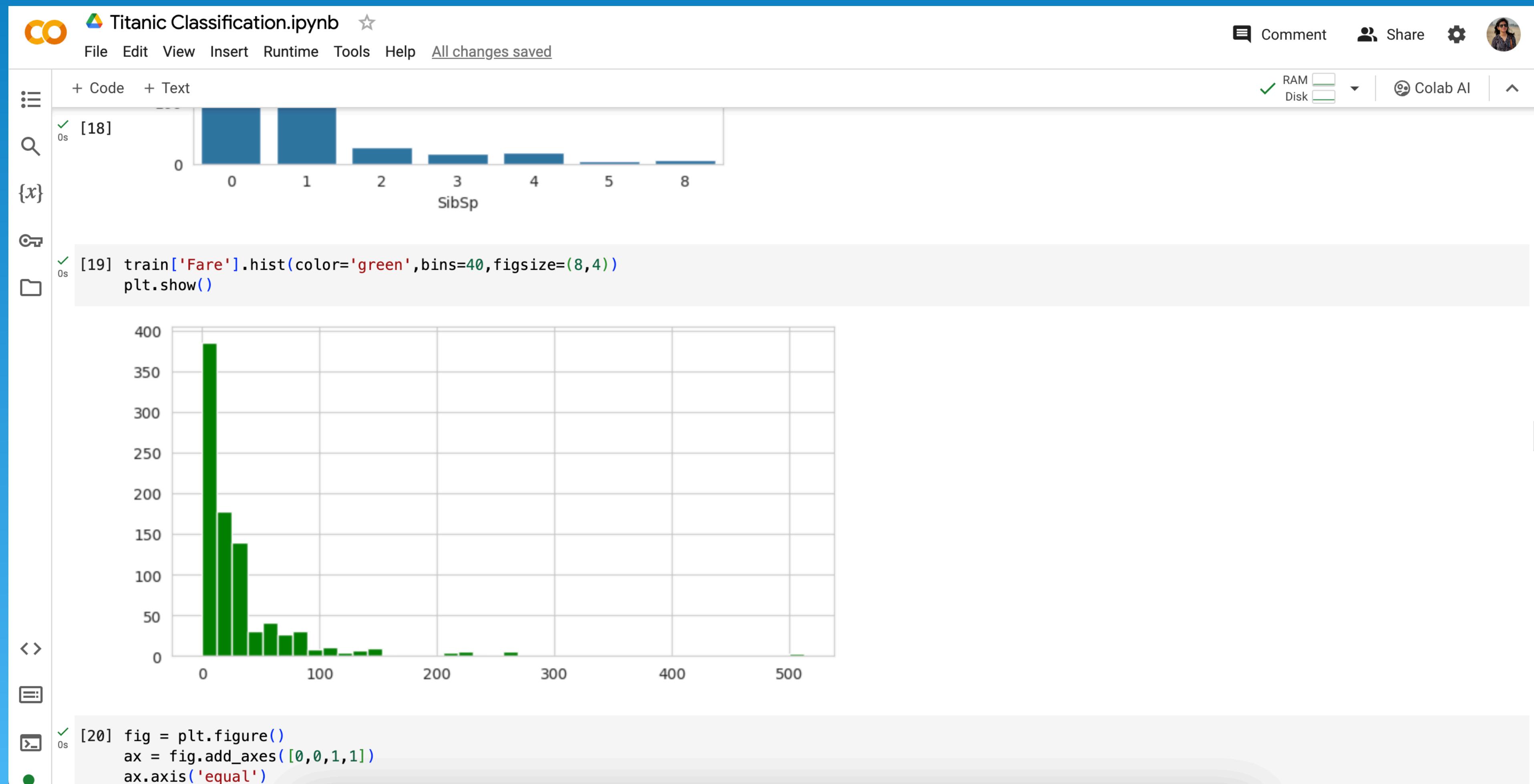
A histogram showing the distribution of passenger ages. The x-axis is labeled 'Age' and ranges from 0 to 80 with major ticks every 10 units. The y-axis represents frequency, ranging from 0 to 50 with major ticks every 10 units. The distribution is unimodal and slightly right-skewed, with the highest frequency occurring around 23 years at approximately 52 individuals. The bars are colored dark red.

[18] sns.countplot(x='SibSp', data=train)
plt.ylabel('Count')
plt.show()

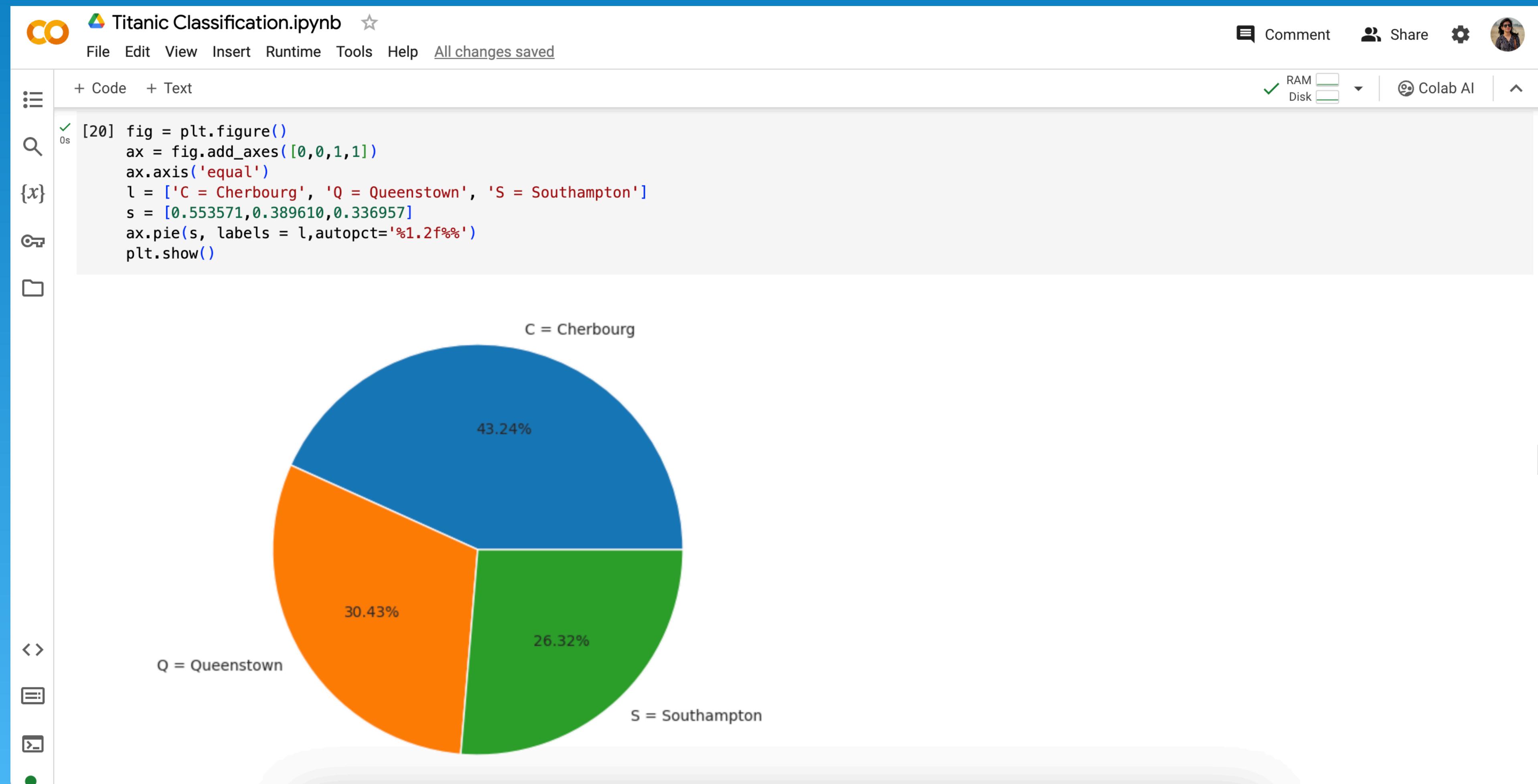
Exploratory Data Analysis for Train data



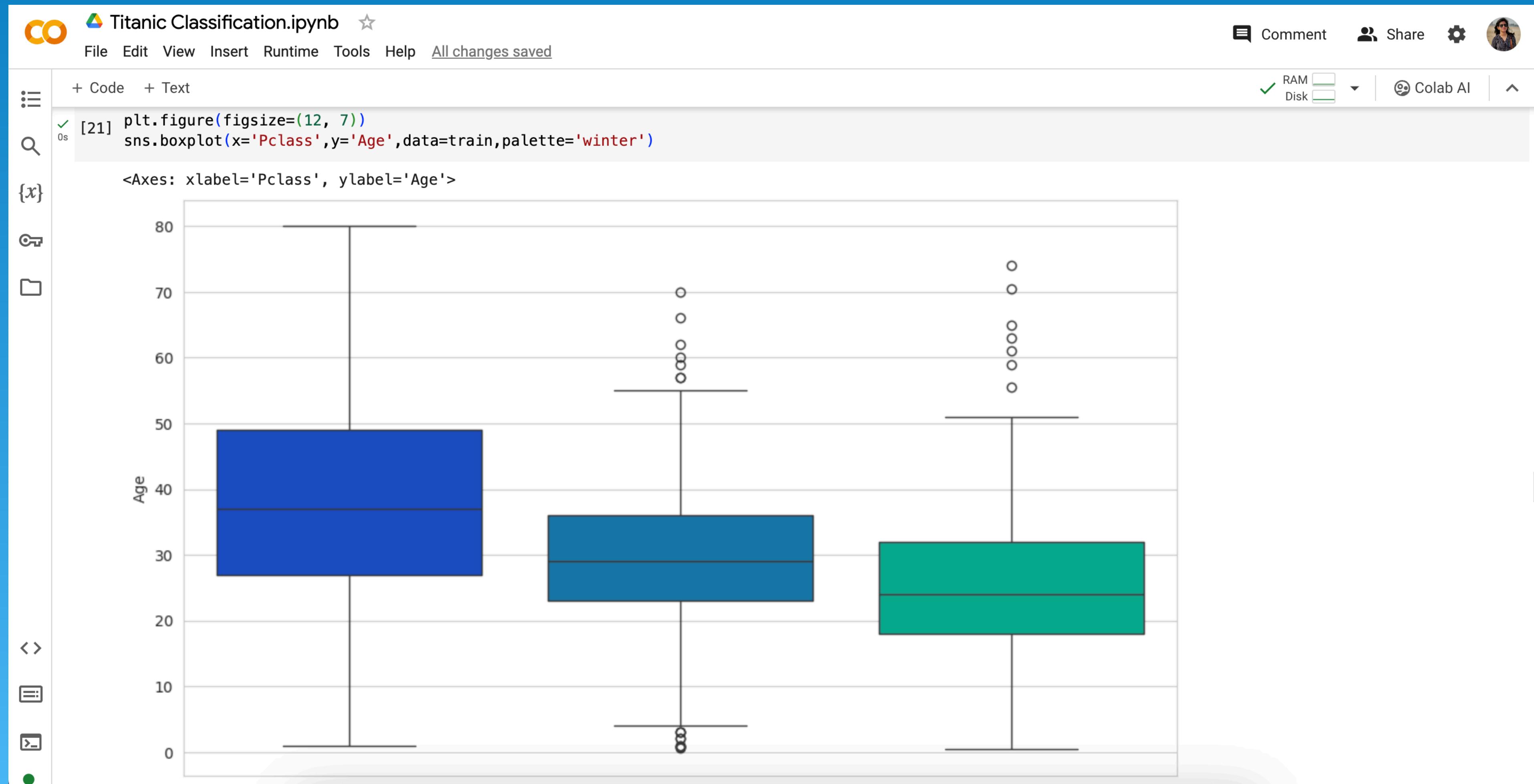
Exploratory Data Analysis for Train data



Exploratory Data Analysis for Train data



Exploratory Data Analysis for Train data



Dropping Useless Columns & Feature Selecting Training values

The screenshot shows a Jupyter Notebook interface with the title "Titanic Classification.ipynb". The notebook contains the following code:

```
[22] train = train.drop(['Ticket'], axis = 1)
      test = test.drop(['Ticket'], axis = 1)

[23] train = train.drop(['Cabin'], axis = 1)
      test = test.drop(['Cabin'], axis = 1)

[24] train = train.drop(['Name'], axis = 1)
      test = test.drop(['Name'], axis = 1)

[25] column_train=['PassengerId','Age','Pclass','SibSp','Parch','Fare','Sex','Embarked']
      X=train[column_train]
      Y=train['Survived']

[26] X['PassengerId'].isnull().sum()
      X['Age'].isnull().sum()
      X['Pclass'].isnull().sum()
      X['SibSp'].isnull().sum()
      X['Parch'].isnull().sum()
      X['Fare'].isnull().sum()
      X['Sex'].isnull().sum()
      X['Embarked'].isnull().sum()
```

The notebook also displays the number "2" at the bottom.

Filling missing values & Converting Categorical to Numerical values

Titanic Classification.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

RAM Disk Colab AI

+ Code + Text

Filling all the missing values

```
[27] X['Age']=X['Age'].fillna(X['Age'].median())
X['Age'].isnull().sum()
```

0

```
[28] X['Embarked'] = train['Embarked'].fillna(method ='pad')
X['Embarked'].isnull().sum()
```

0

Converting Categorical into Numerical values

```
[29] d={'male':0, 'female':1}
X['Sex'] = X['Sex'].apply(lambda x:d[x])
X['Sex'].head()
```

0	0
1	1
2	1
3	1
4	0

Name: Sex, dtype: int64

```
[30] e={'C':0, 'Q':1 , 'S':2}
X['Embarked']=X['Embarked'].apply(lambda x:e[x])
X['Embarked'].head()
```

0	2
---	---

Cross-checking for null values and its datatypes

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** Titanic Classification.ipynb
- Cells:**
 - [30] e={'C':0, 'Q':1, 'S':2}
X['Embarked']=X['Embarked'].apply(lambda x:e[x])
X['Embarked'].head()
Output:
0 2
1 0
2 2
3 2
4 2
Name: Embarked, dtype: int64
 - [31] X.info()
Output:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
 # Column Non-Null Count Dtype

 0 PassengerId 891 non-null int64
 1 Age 891 non-null float64
 2 Pclass 891 non-null int64
 3 SibSp 891 non-null int64
 4 Parch 891 non-null int64
 5 Fare 891 non-null float64
 6 Sex 891 non-null int64
 7 Embarked 891 non-null int64
dtypes: float64(2), int64(6)
memory usage: 55.8 KB
 - [32] sns.heatmap(X.isnull(),yticklabels=False,cbar=False,cmap='viridis')
Output:
<Axes: >
- Toolbar:** Comment, Share, Settings, User Profile
- Runtime Status:** RAM and Disk usage indicators

Heatmap for Train set to cross-check null values & Training Testing Splitting the model

Titanic Classification.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

RAM Disk Colab AI

+ Code + Text

0s

```
0s ➔ sns.heatmap(X.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

{x}

<Axes: >

PassengerId Age Pclass SibSp Parch Fare Sex Embarked

<>

Training Testing and Splitting the model

0s

```
[36] from sklearn.model_selection import train_test_split
     X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3,random_state=7)
```

Train-Test Split using Logistic Regression

Titanic Classification.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

RAM Disk Colab AI

+ Code + Text

[36] from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3,random_state=7)

{x} Using Logistic Regression

[37] from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,Y_train)
Y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,Y_pred))

Accuracy Score: 0.7686567164179104

[38] #Confusion Matrix & Classification Report
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
confusion_mat = confusion_matrix(Y_test,Y_pred)
print(confusion_mat)
print(classification_report(Y_test,Y_pred))

[[132 24]	[38 74]]		
precision	recall	f1-score	support
0 0.78	0.85	0.81	156
1 0.76	0.66	0.70	112
accuracy		0.77	268
macro avg	0.77	0.75	0.76
weighted avg	0.77	0.77	0.77

Train-Test Split using Support Vector Machine

Titanic Classification.ipynb 

File Edit View Insert Runtime Tools Help All changes saved

Comment Share  

+ Code + Text  RAM  Disk Colab AI ^

Using Support Vector Machine

```
[39] from sklearn.svm import SVC
     model1 = SVC()
     model1.fit(X_train,Y_train)
     Y_pred1 = model1.predict(X_test)

     from sklearn.metrics import accuracy_score
     print("Acc=",accuracy_score(Y_test,Y_pred1))

     Acc= 0.6156716417910447
```

```
[40] #Confusion Matrix & Classification Report
     from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
     confusion_mat = confusion_matrix(Y_test,Y_pred1)
     print(confusion_mat)
     print(classification_report(Y_test,Y_pred1))

[[155  1]
 [102  10]]
      precision    recall  f1-score   support
          0       0.60      0.99      0.75      156
          1       0.91      0.09      0.16      112
   accuracy                           0.62      268
  macro avg       0.76      0.54      0.46      268
weighted avg       0.73      0.62      0.50      268
```



Train-Test Split using K-Nearest Neighbors

CO Titanic Classification.ipynb ★

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙️ Colab AI ^

+ Code + Text RAM Disk

Using K -Nearest Neighbors

```
[41] from sklearn.neighbors import KNeighborsClassifier
    model2 = KNeighborsClassifier(n_neighbors=5)
    model2.fit(X_train,Y_train)
    Y_pred2 = model2.predict(X_test)

    from sklearn.metrics import accuracy_score
    print("Accuracy Score:",accuracy_score(Y_test,Y_pred2))

Accuracy Score: 0.5783582089552238
```

```
[42] #Confusion Matrix & Classification Report
    from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
    confusion_mat = confusion_matrix(Y_test,Y_pred2)
    print(confusion_mat)
    print(classification_report(Y_test,Y_pred2))

[[120  36]
 [ 77  35]]
      precision    recall   f1-score   support
          0       0.61      0.77      0.68      156
          1       0.49      0.31      0.38      112

      accuracy         0.55      0.54      0.53      268
      macro avg       0.55      0.54      0.53      268
      weighted avg    0.56      0.58      0.56      268
```

Train-Test Split using Naive Bayes Classifier

CO Titanic Classification.ipynb ★

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙️ Colab AI ^

+ Code + Text RAM Disk

Using Naive Bayes Classifier

```
[43]: from sklearn.naive_bayes import GaussianNB
model3 = GaussianNB()
model3.fit(X_train,Y_train)
Y_pred3 = model3.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,Y_pred3))
```

Accuracy Score: 0.7686567164179104

```
[44]: #Confusion Matrix & Classification Report
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
confusion_mat = confusion_matrix(Y_test,Y_pred3)
print(confusion_mat)
print(classification_report(Y_test,Y_pred3))
```

[[129 27]	[35 77]]
	precision recall f1-score support
0	0.79 0.83 0.81 156
1	0.74 0.69 0.71 112
	accuracy 0.77 268
macro avg	0.76 0.76 0.76 268
weighted avg	0.77 0.77 0.77 268

Train-Test Split using Decision Tree

Titanic Classification.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

RAM Disk Colab AI

Using Decision Tree

```
[45]: from sklearn.tree import DecisionTreeClassifier
model4 = DecisionTreeClassifier(criterion='entropy',random_state=7)
model4.fit(X_train,Y_train)
Y_pred4 = model4.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,Y_pred4))

Accuracy Score: 0.746268656716418
```

```
[46]: #Confusion Matrix & Classification Report
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
confusion_mat = confusion_matrix(Y_test,Y_pred4)
print(confusion_mat)
print(classification_report(Y_test,Y_pred4))

[[128 28]
 [ 40 72]]
      precision    recall  f1-score   support
          0       0.76      0.82      0.79      156
          1       0.72      0.64      0.68      112
   accuracy                           0.75      268
  macro avg       0.74      0.73      0.73      268
weighted avg       0.74      0.75      0.74      268
```

results = pd.DataFrame([

Comparing All used ML Models

Titanic Classification.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings User profile

RAM Disk Colab AI

+ Code + Text

[46] 0s accuracy 0.75 268
macro avg 0.74 0.73 268
weighted avg 0.74 0.75 0.74 268

[47] results = pd.DataFrame({
 'Model': ['Logistic Regression', 'Support Vector Machines', 'K -Nearest Neighbors', 'Naive Bayes Classifier', 'Decision Tree'],
 'Score': [0.77, 0.62, 0.58, 0.77, 0.75]})

result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
result_df.head(9)

Model	Score
Logistic Regression	0.77
Naive Bayes Classifier	0.77
Decision Tree	0.75
Support Vector Machines	0.62
K -Nearest Neighbors	0.58

Make predictions on the test set

[48] #Using Logistic Regression Model
test_predictions = model.predict(X_test)

Making Predictions on the highest F1 score Model(Logistic Regression) & saving in a csv.file

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** Titanic Classification.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved.
- Right Sidebar:** Comment, Share, Settings, RAM/Disk status, Colab AI, and a user profile icon.
- Code Cells:**
 - [48] `#Using Logistic Regression Model`
test_predictions = model.predict(X_test)
 - [49] `0s print("Length of 'PassengerId':", len(test['PassengerId']))`
`print("Length of 'Survived':", len(test_predictions))`
Length of 'PassengerId': 418
Length of 'Survived': 268
 - [50] `0s # Align the lengths by taking the first N rows where N is the minimum length`
`min_length = min(len(test['PassengerId']), len(test_predictions))`
`submission_df = pd.DataFrame({`
`'PassengerId': test['PassengerId'].head(min_length),`
`'Survived': test_predictions[:min_length]`
})
 - [51] `0s submission_df.to_csv('submission_logistic_regression.csv', index=False)`
 - [52] `0s submission_df.head(10)`
- Output:** The output for cell [52] shows the first 10 rows of the submission DataFrame:

PassengerId	Survived
1	0
2	1
3	1
4	0
5	0
6	1
7	1
8	0
9	1
10	0

Fetching the head of newly created dataset

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** Titanic Classification.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved
- Code Cells:**
 - [51] `submission_df.to_csv('submission_logistic_regression.csv', index=False)`
 - [52] `submission_df.head(10)`
- Data Preview:** The output of cell [52] is a table showing the first 10 rows of the `submission_df`.

	PassengerId	Survived
0	892	0
1	893	0
2	894	0
3	895	0
4	896	1
5	897	1
6	898	0
7	899	0
8	900	0
9	901	1
- Sidebar:** Includes icons for Code, Text, Comment, Share, and Colab AI.

Thank you