

GROUP-1 TITANIC DATASET

**NAME:
KHUSHBOO
NAVYA
HARSHA**

PROBLEM STATEMENT:

**TO FIND SURVIVAL STATUS OF
PASSENGER ON THE TITANIC.**

```
+ Code + Text Copy to Clipboard
0 Load dataset
titanic_df = pd.read_csv('content/titanic.csv')

1 Exploratory Data Analysis (EDA)
print(titanic_df.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 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        891 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  891 non-null object
12 FamilySize 891 non-null int64
dtypes: float64(2), int64(7), object(4)
memory usage: 98.6+ KB
None
```

```
0 Load dataset
titanic_df = pd.read_csv('content/titanic.csv')

1 Exploratory Data Analysis (EDA)
print(titanic_df.info())

2 Print the first 5 rows of the dataset
print(titanic_df.head())

3 Print the last 5 rows of the dataset
print(titanic_df.tail())

4 Print the data types of the columns
print(titanic_df.dtypes)

5 Print the number of non-null values for each column
print(titanic_df.isnull().sum())

6 Print the number of unique values for each column
print(titanic_df.nunique())

7 Print the correlation matrix
print(titanic_df.corr())

8 Print the correlation matrix as a heatmap
plt.figure(figsize=(10,8))
sns.heatmap(titanic_df.corr(),cmap=cm.RdBu,annot=True)
plt.show()
```

```
0 Load dataset
titanic_df = pd.read_csv('content/titanic.csv')

1 Exploratory Data Analysis (EDA)
print(titanic_df.info())

2 Print the first 5 rows of the dataset
print(titanic_df.head())

3 Print the last 5 rows of the dataset
print(titanic_df.tail())

4 Print the data types of the columns
print(titanic_df.dtypes)

5 Print the number of non-null values for each column
print(titanic_df.isnull().sum())

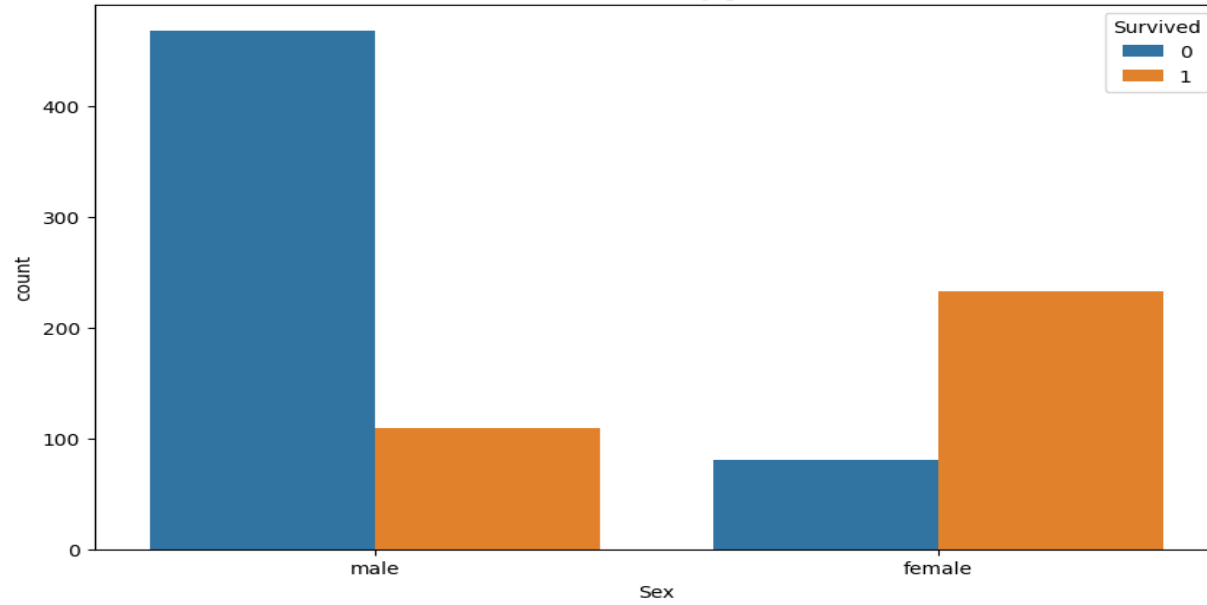
6 Print the number of unique values for each column
print(titanic_df.nunique())

7 Print the correlation matrix
print(titanic_df.corr())

8 Print the correlation matrix as a heatmap
plt.figure(figsize=(10,8))
sns.heatmap(titanic_df.corr(),cmap=cm.RdBu,annot=True)
plt.show()
```

PRINTING HEAD, DESCRIPTION AND INFORMATION OF TITANIC DATASET

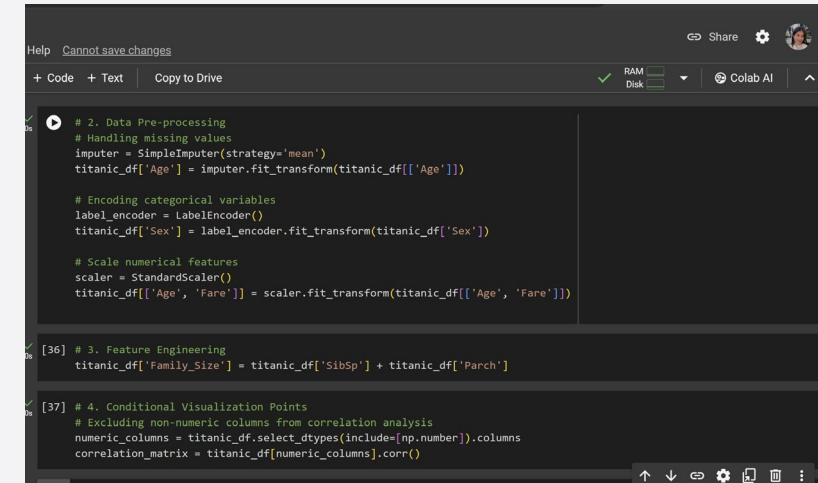
Survival count by gender



PAIRPLOT AND COUNTPLOT :-



- HANDLING MISSING VALUES, ENCODING CATEGORICAL VARIABLES AND SCALE NUMERICAL FEATURES:-
- FEATURE ENGINEERING:
- CONDITIONAL VISUALIZATION POINTS:-



```
Help Cannot save changes
+ Code + Text Copy to Drive RAM Disk Colab AI
# 2. Data Pre-processing
# Handling missing values
imputer = SimpleImputer(strategy='mean')
titanic_df['Age'] = imputer.fit_transform(titanic_df[['Age']])

# Encoding categorical variables
label_encoder = LabelEncoder()
titanic_df['Sex'] = label_encoder.fit_transform(titanic_df['Sex'])

# Scale numerical features
scaler = StandardScaler()
titanic_df[['Age', 'Fare']] = scaler.fit_transform(titanic_df[['Age', 'Fare']])

[36] # 3. Feature Engineering
titanic_df['Family_Size'] = titanic_df['SibSp'] + titanic_df['Parch']

[37] # 4. Conditional Visualization Points
# Excluding non-numeric columns from correlation analysis
numeric_columns = titanic_df.select_dtypes(include=[np.number]).columns
correlation_matrix = titanic_df[numeric_columns].corr()
```

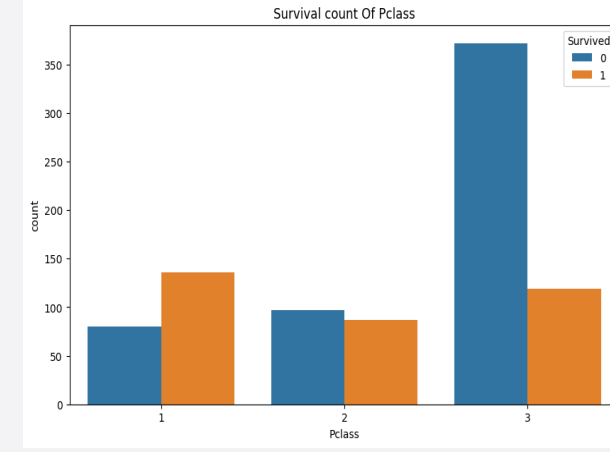
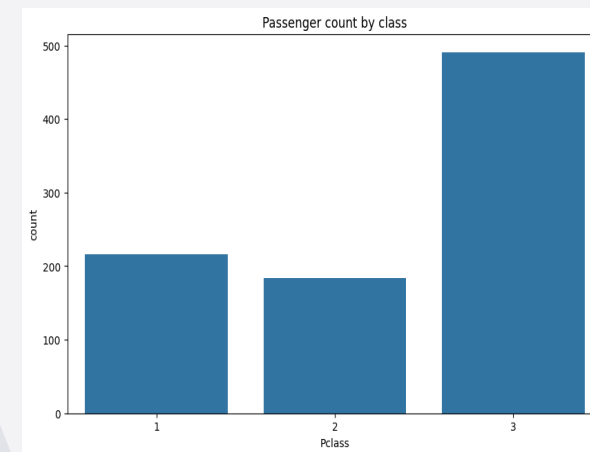
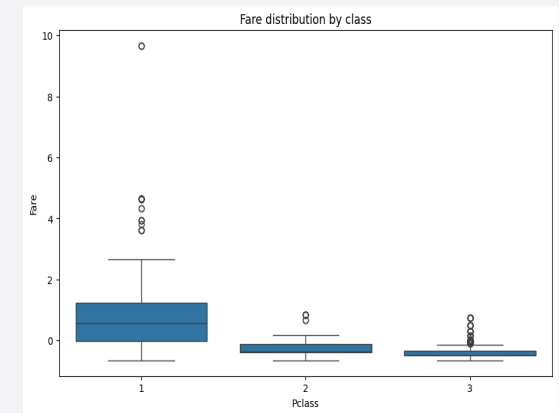
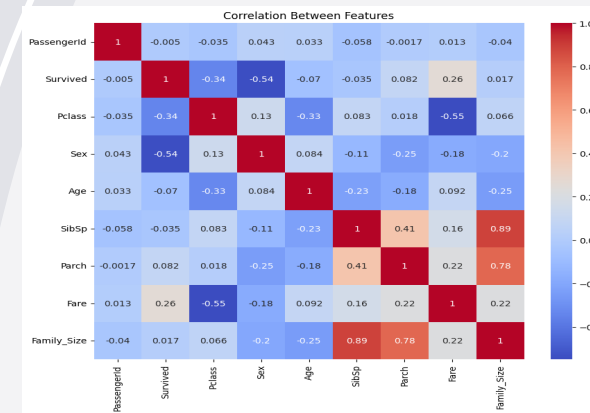
HEATMAP ,BOXPLOT AND COUNTPLOTS:

PLOTTING THE HEATMAP WITH
CORRELATION BETWEEN
FEATURES.

PLOTTING BOXPLOT FOR FARE
DISTRIBUTION BY CLASS.

PLOTTING COUNTPLOT FOR
PASSENGER COUNT BY CLASS.

PLOTTING COUNTPLOT OF
SURVIVAL COUNT OF PCLASS.



```
✓ [25] # Count of Non-Surviving Passengers Between Ages 20-30
0s
young_non_survivors = titanic_df[(titanic_df['Age'] >= 20) & (titanic_df['Age'] <= 30) & (titanic_df['Survived'] == 0)
print("Count of non-surviving passengers between ages 20-30:", young_non_survivors)

Count of non-surviving passengers between ages 20-30: 0
```

```
Help Cannot save changes
+ Code + Text Copy to Drive
✓ [39] # 5. Models for Analysis
0s
X = titanic_df.drop(['Survived', 'Name', 'Ticket', 'Cabin', 'Embarked'], axis=1)
y = titanic_df['Survived']

✓ [40] # Splitting data into train and test sets
0s
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

✓ [41] # Initializing models
0s
models = {
    "Logistic Regression": LogisticRegression(),
    "K Nearest Neighbors": KNeighborsClassifier(),
    "Support Vector Classifier": SVC(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier()
}
```

COUNTING, ANALYZING, SPLITTING AND INITIALIZING.

INITIALIZING, SCALING, TRAINING AND EVALUATING MODELS

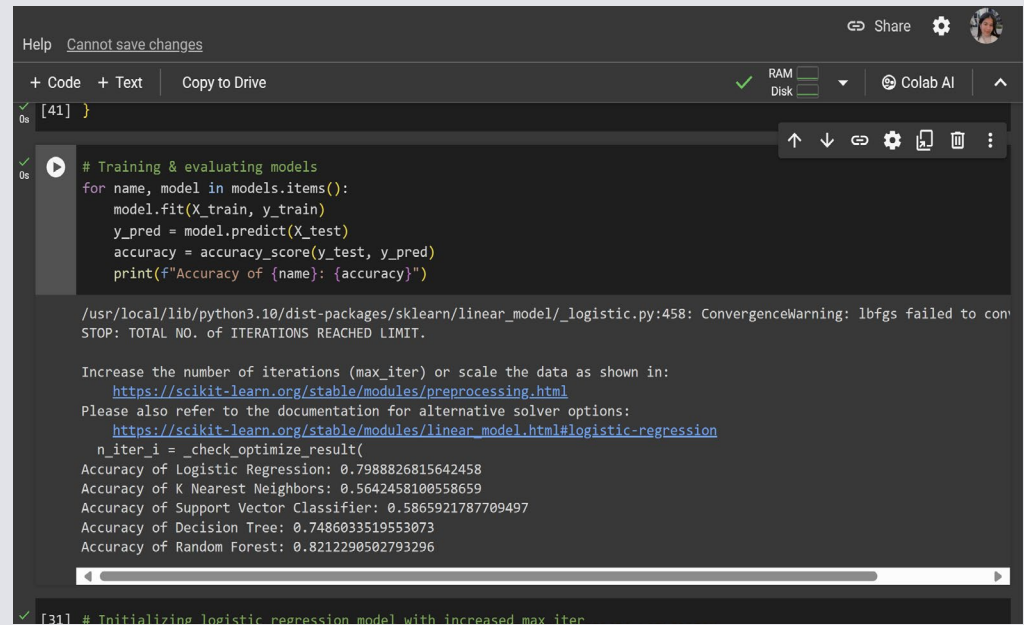
```
[31] # Initializing logistic regression model with increased max_iter
logistic_model = LogisticRegression(max_iter=1000)

# Scaling the numerical features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Training the model on scaled data
logistic_model.fit(X_train_scaled, y_train)

# Evaluating the model
y_pred = logistic_model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of Logistic Regression: {accuracy}")

Accuracy of Logistic Regression: 0.8100558659217877
```



The screenshot shows a Google Colab notebook interface. At the top, there's a 'Help' button and a 'Cannot save changes' message. Below that, a toolbar includes '+ Code', '+ Text', 'Copy to Drive', and status indicators for RAM and Disk. The main code cell, labeled '[41]', contains a loop that trains and evaluates several models. The output shows a 'ConvergenceWarning' from sklearn's lbfgs solver, indicating that the maximum number of iterations was reached. Below the warning, a list of accuracies for various models is displayed. The bottom of the notebook shows the start of the next cell, '[31]', which begins with initializing a logistic regression model.

```
[41] # Training & evaluating models
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy of {name}: {accuracy}")

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression
n_iter_i = _check_optimize_result(
Accuracy of Logistic Regression: 0.7988826815642458
Accuracy of K Nearest Neighbors: 0.5642458100558659
Accuracy of Support Vector Classifier: 0.5865921787709497
Accuracy of Decision Tree: 0.7486033519553073
Accuracy of Random Forest: 0.8212290502793296

[31] # Initializing logistic regression model with increased max_iter
```


CONCLUSION:-

Here by survival by Gender we have analysed that Females survival Count is more than Male.

Non-Survival count for Class 3 is greater than class 1 and class 2.

Overall, a box plot analysis on fare by passenger class in the Titanic dataset offers a visual summary of fare distributions within each class and facilitates comparisons between classes, enabling deeper insights into the socioeconomic dynamics among Titanic passengers.

The count of passengers are way higher in 3rd class followed by 1st class and 2nd class.

**Count of non-surviving passengers between ages 20-30: 0
Random Forest has good accuracy compared to other algorithms in this project.**

The background is a light gray gradient. It features several 3D plus signs (+) of varying sizes and colors. Some are white with gray shadows, some are dark gray, and two are a vibrant blue. These plus signs are scattered across the frame, with a higher concentration on the right side. A large, thin white circle is positioned in the lower-left quadrant, partially overlapping the text.

THANK YOU