**DATASET: TITANIC**

**Step 1: Data Collection and Preprocessing**

1. **Load the Titanic Dataset**:
   * Download the Titanic dataset from Kaggle and load it into your environment.
2. **Handle Missing Data**:
   * Inspect columns for missing data (e.g., Age, Embarked, Cabin).
   * Decide how to handle missing values. For example:
     + For numerical data (like Age), you can fill the missing values with the mean or median.
     + For categorical data (like Embarked), you can fill missing values with the most frequent category (mode).
     + Drop unnecessary columns like Cabin, which might not provide useful insights or contain too many missing values.
3. **Feature Engineering**:
   * **Create new features** (optional): For example, you can create an IsChild feature by checking if the Age is under a specific value (e.g., 18).
   * **Convert categorical features into numeric**:
     + Use One-Hot Encoding or Label Encoding for features like Sex, Embarked, etc.
     + Example: Convert Sex to 0 (male) and 1 (female).
4. **Remove Irrelevant Columns**:
   * Drop columns that are not needed, like Cabin, since you decided to remove them earlier.

**Step 2: Exploratory Data Analysis (EDA)**

1. **Descriptive Statistics**:
   * Calculate summary statistics for numeric columns like Age, Fare, Pclass, etc.
   * Display values like mean, median, standard deviation, minimum, and maximum.
2. **Univariate Analysis (Distributions of Variables)**:
   * Plot **histograms** for numerical features like Age and Fare to see their distribution.
   * Plot **bar charts** for categorical features like Pclass, Survived, Sex, and Embarked.
3. **Bivariate Analysis (Relationships Between Variables)**:
   * Use **box plots** or **violin plots** to show the distribution of variables like Fare or Age across different Survived categories.
   * Plot **correlation heatmaps** to analyze how features are correlated with each other and with the target variable (Survived).
   * Example:
     + Survival rates by passenger class.
     + Survival rates by gender.

**Step 3: Data Visualization**

**Tableau**:

1. **Load Data into Tableau**:
   * Import the cleaned dataset into Tableau.
2. **Create Visualizations**:
   * Create interactive dashboards with visualizations like:
     + Pie chart or bar chart showing the percentage of survivors vs. non-survivors.
     + Bar charts visualizing survival rates across Pclass, Sex, or Age groups.
     + A heatmap to analyze survival rates across multiple dimensions (e.g., age and class).
   * Use filters and interactive options to make the visualization more dynamic.

**Power BI**:

1. **Load Data into Power BI**:
   * Import the cleaned dataset into Power BI.
2. **Create Visualizations**:
   * Similar to Tableau, create:
     + Bar charts showing survival rates by class, gender, and age group.
     + Pie chart showing the total number of survivors vs. non-survivors.
   * Customize reports with slicers to explore different attributes.

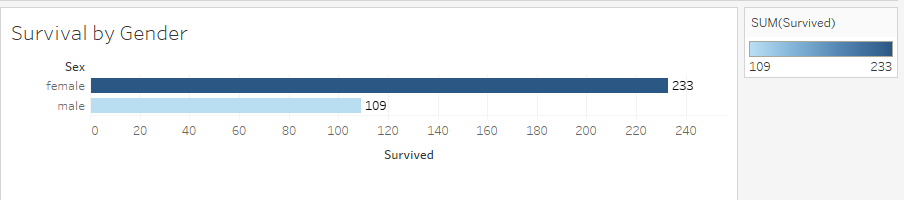
**Step 4: Machine Learning Model Building and Algorithm Comparison**

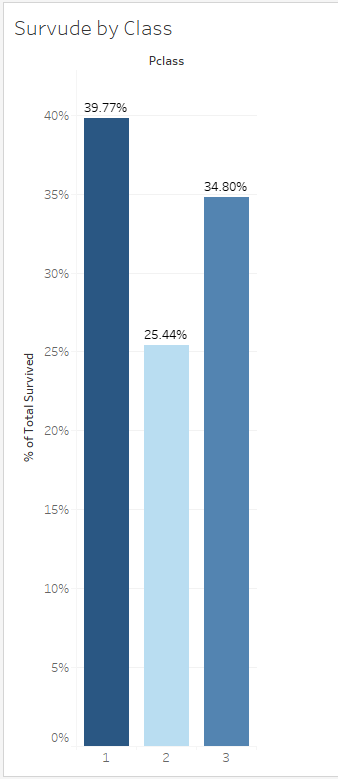
1. **Split Data**:
   * Split your data into training and testing datasets (e.g., 80% training, 20% testing).
2. **Model Selection**:
   * **Logistic Regression**: Start with a basic logistic regression model to predict the survival rate.
   * **Random Forest Classifier**: Try more complex models like Random Forest for better predictions.
   * **Support Vector Machine (SVM)**: Compare it with an SVM model.
3. **Model Training**:
   * Train the models using the training data (X\_train, y\_train).
   * Evaluate the model on test data (X\_test, y\_test).
4. **Model Evaluation**:
   * Evaluate the models based on performance metrics:
     + **Accuracy**: Percentage of correctly predicted instances.
     + **Precision**: How many of the predicted positives are actually positive.
     + **Recall**: How many of the actual positives were correctly identified.
     + **F1-Score**: Harmonic mean of precision and recall.
   * Generate **confusion matrix** and **ROC curve** to visualize model performance.

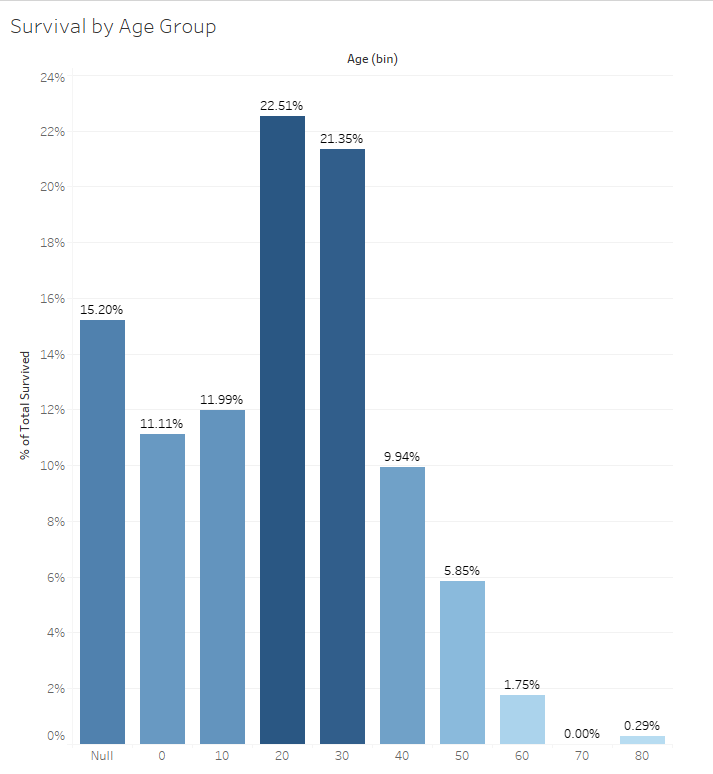
**Step 5: Comparative Analysis of the Models**

1. **Compare the Performance**:
   * Check the metrics (Accuracy, Precision, Recall, F1-Score) for each model.
   * Use **cross-validation** to confirm the robustness of your models.
2. **Discussion**:
   * Highlight the best-performing model.
   * Discuss the reasons behind the performance, such as how well the features explain survival.
   * Discuss trade-offs between model complexity and performance.

**IN TBALEAU**

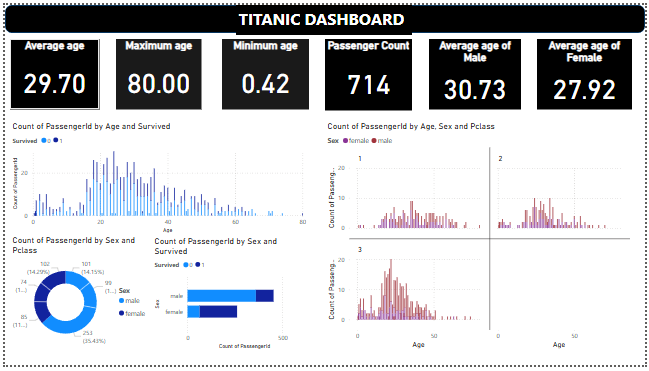








IN POWER BI:



So in power bi we use these above charts and graphs to show the visualization of our datasets titanic from Kaggle source which includes average age, maximum age, minimum age, passenger counts, average age of males, average age of females.

NOW FOR COMPARISON WE used different algorithms to apply on this dataset:

**1.DECISION TREE:**

#---DECISION TREE-------

# Import necessary libraries for Keras and TensorFlow

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# Load the Titanic dataset from the given path

data = pd.read\_csv("/kaggle/input/titanic-dataset/Titanic-Dataset.csv")

# Encode categorical variables

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

data['Embarked'] = data['Embarked'].fillna('S')

data['Embarked'] = data['Embarked'].map({'S': 0, 'C': 1, 'Q': 2})

# Drop irrelevant columns

data = data.drop(['Name', 'Ticket', 'Cabin'], axis=1)

# Handle missing values in 'Age'

data['Age'] = data['Age'].fillna(data['Age'].median())

X = data.drop('Survived', axis=1)

y = data['Survived']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = DecisionTreeClassifier(max\_depth=3, random\_state=42)

model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

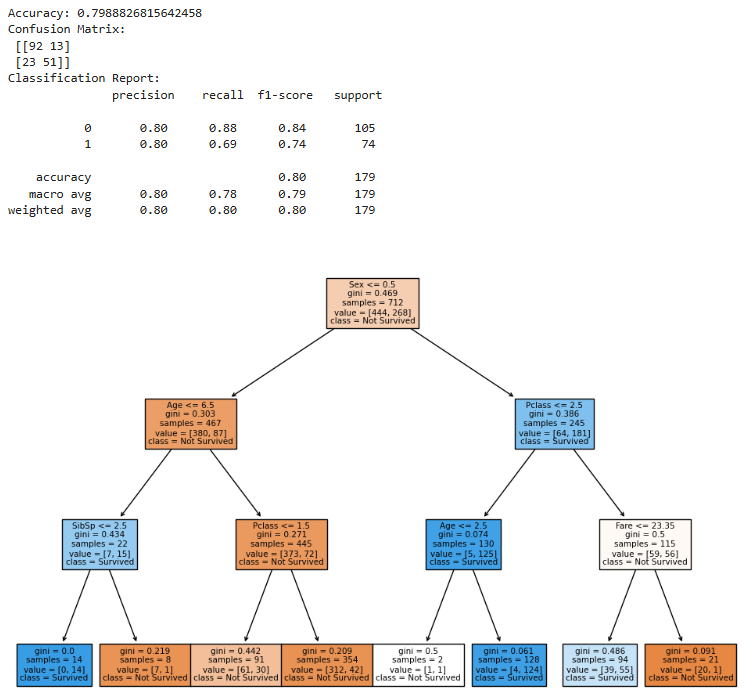
from sklearn.tree import plot\_tree

import matplotlib.pyplot as plt

plt.figure(figsize=(12, 8))

plot\_tree(model, feature\_names=X.columns, class\_names=['Not Survived', 'Survived'], filled=True)

plt.show()



**2.RANDOM FOREST:**

#--------------------RANDOM FOREST-----------------

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# Load the Titanic dataset from the given path

data = pd.read\_csv("/kaggle/input/titanic-dataset/Titanic-Dataset.csv")

# Encode categorical variables

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

data['Embarked'] = data['Embarked'].fillna('S')

data['Embarked'] = data['Embarked'].map({'S': 0, 'C': 1, 'Q': 2})

# Drop irrelevant columns

data = data.drop(['Name', 'Ticket', 'Cabin'], axis=1)

# Fill missing 'Age' with median

data['Age'] = data['Age'].fillna(data['Age'].median())

# Define features and target

X = data.drop('Survived', axis=1)

y = data['Survived']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize the Random Forest model

rf\_model = RandomForestClassifier(n\_estimators=100, max\_depth=5, random\_state=42)

# Train the model

rf\_model.fit(X\_train, y\_train)

# Make predictions

y\_pred = rf\_model.predict(X\_test)

# Accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

# Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:\n", cm)

# Classification Report

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

# Feature importance

import matplotlib.pyplot as plt

feature\_importance = pd.Series(rf\_model.feature\_importances\_, index=X.columns).sort\_values(ascending=False)

# Plot feature importance

plt.figure(figsize=(8, 6))

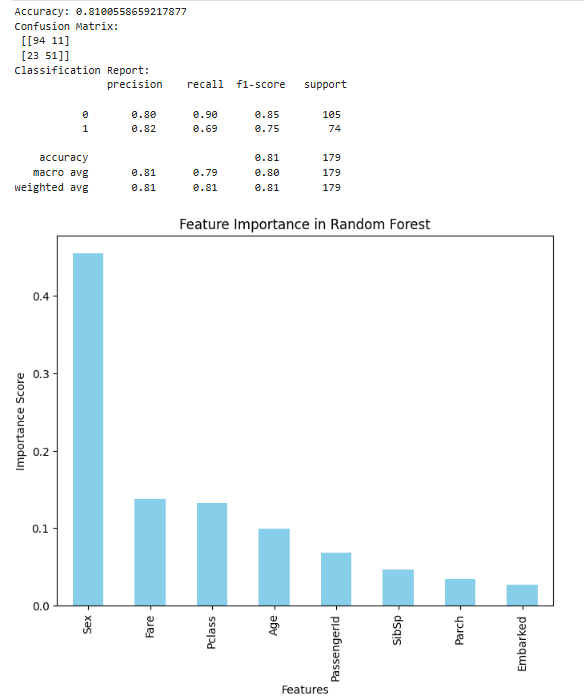
feature\_importance.plot(kind='bar', color='skyblue')

plt.title("Feature Importance in Random Forest")

plt.xlabel("Features")

plt.ylabel("Importance Score")

plt.show()



**3. LOGISTIC REGRESSION:**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# Load the Titanic dataset from the given path

data = pd.read\_csv("/kaggle/input/titanic-dataset/Titanic-Dataset.csv")

# Encode categorical variables

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

data['Embarked'] = data['Embarked'].fillna('S')

data['Embarked'] = data['Embarked'].map({'S': 0, 'C': 1, 'Q': 2})

# Drop irrelevant columns

data = data.drop(['Name', 'Ticket', 'Cabin'], axis=1)

# Handle missing values

data['Age'] = data['Age'].fillna(data['Age'].median())

# Define features and target variable

X = data.drop('Survived', axis=1)

y = data['Survived']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize Logistic Regression model

log\_model = LogisticRegression(max\_iter=1000, random\_state=42)

# Train the model

log\_model.fit(X\_train, y\_train)

# Make predictions

y\_pred = log\_model.predict(X\_test)

# Accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

# Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:\n", cm)

# Classification Report

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

# Feature importance

coefficients = pd.Series(log\_model.coef\_[0], index=X.columns).sort\_values(ascending=False)

# Display coefficients

print("Feature Coefficients:\n", coefficients)

# Plot feature importance

import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))

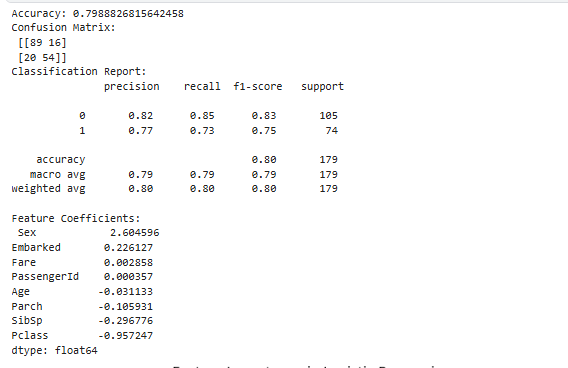
coefficients.plot(kind='bar', color='green')

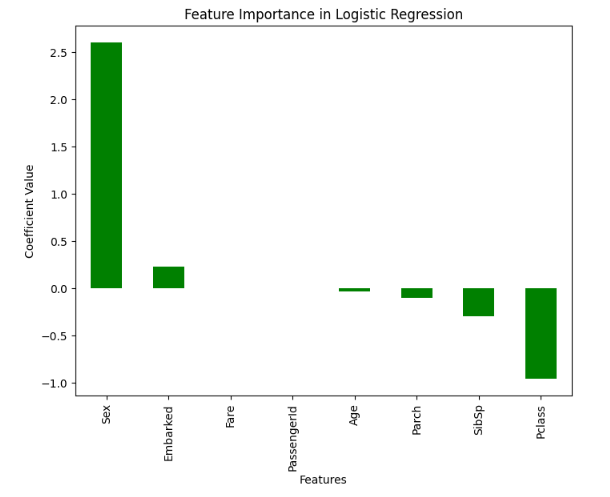
plt.title("Feature Importance in Logistic Regression")

plt.xlabel("Features")

plt.ylabel("Coefficient Value")

plt.show()





**4. NETURAL NETWORK:**

#-----------NETURAL NETWORK ---------------------

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# Load the Titanic dataset from the given path

data = pd.read\_csv("/kaggle/input/titanic-dataset/Titanic-Dataset.csv")

# Encode categorical variables

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

data['Embarked'] = data['Embarked'].fillna('S')

data['Embarked'] = data['Embarked'].map({'S': 0, 'C': 1, 'Q': 2})

# Drop irrelevant columns

data = data.drop(['Name', 'Ticket', 'Cabin'], axis=1)

# Fill missing 'Age' with median

data['Age'] = data['Age'].fillna(data['Age'].median())

# Define features and target variable

X = data.drop('Survived', axis=1)

y = data['Survived']

# Split the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Scale the features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Initialize the model

model = Sequential()

# Input layer and hidden layers

model.add(Dense(16, input\_dim=X\_train.shape[1], activation='relu')) # First hidden layer

model.add(Dropout(0.2)) # Dropout to prevent overfitting

model.add(Dense(8, activation='relu')) # Second hidden layer

model.add(Dense(1, activation='sigmoid')) # Output layer (sigmoid for binary classification)

# Compile the model

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Train the model

history = model.fit(X\_train, y\_train, validation\_split=0.2, epochs=50, batch\_size=32, verbose=1)

# Evaluate on test data

y\_pred = (model.predict(X\_test) > 0.5).astype("int32")

# Accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

# Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:\n", cm)

# Classification Report

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

import matplotlib.pyplot as plt

# Plot loss

plt.plot(history.history['loss'], label='Training Loss')

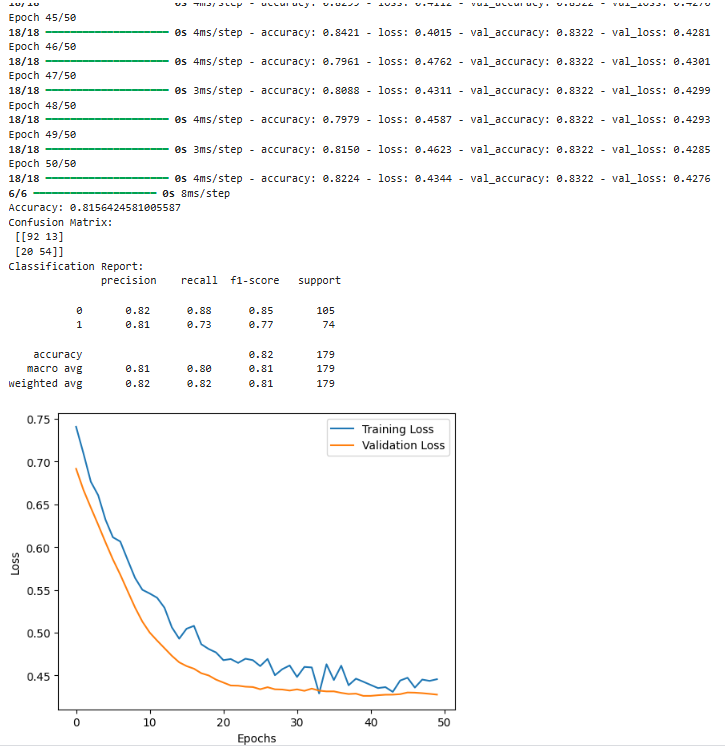
plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()



**5.Support Vector Machine (SVM):**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# Load the Titanic dataset from the given path

data = pd.read\_csv("/kaggle/input/titanic-dataset/Titanic-Dataset.csv")

# Encode categorical variables

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

data['Embarked'] = data['Embarked'].fillna('S')

data['Embarked'] = data['Embarked'].map({'S': 0, 'C': 1, 'Q': 2})

# Drop irrelevant columns

data = data.drop(['Name', 'Ticket', 'Cabin'], axis=1)

# Fill missing 'Age' with median

data['Age'] = data['Age'].fillna(data['Age'].median())

# Define features and target variable

X = data.drop('Survived', axis=1)

y = data['Survived']

# Split the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Scale the features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Initialize the SVM model

svm\_model = SVC(kernel='rbf', C=1, gamma='scale', random\_state=42)

# Train the SVM model

svm\_model.fit(X\_train, y\_train)

# Make predictions

y\_pred = svm\_model.predict(X\_test)

# Accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

# Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:\n", cm)

# Classification Report

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

from sklearn.model\_selection import GridSearchCV

# Define parameter grid

param\_grid = {

'C': [0.1, 1, 10, 100],

'gamma': ['scale', 'auto', 0.1, 1, 10],

'kernel': ['linear', 'rbf', 'poly']

}

# Perform grid search

grid\_search = GridSearchCV(SVC(random\_state=42), param\_grid, cv=5, scoring='accuracy', verbose=1)

grid\_search.fit(X\_train, y\_train)

# Best parameters and model

print("Best Parameters:", grid\_search.best\_params\_)

best\_svm\_model = grid\_search.best\_estimator\_

