* Name: Dipak Shingade
* Roll No.:21102B0029

<https://github.com/Dipakgith/Ml-lab-expriments>

import pandas as pd import numpy as np

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

from sklearn.preprocessing import OneHotEncoder, StandardScaler from sklearn.impute import SimpleImputer

from sklearn.compose import ColumnTransformer from sklearn.pipeline import Pipeline

from sklearn.linear\_model import LogisticRegression

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

import matplotlib.pyplot as plt import seaborn as sns

*# Load the dataset*

train\_data = pd.read\_csv('train.csv')

*# Select features for the model*

features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']

X = train\_data[features]

y = train\_data['Survived']

*# Create preprocessing pipelines for both numeric and categorical data* numeric\_features = ['Age', 'SibSp', 'Parch', 'Fare'] categorical\_features = ['Pclass', 'Sex', 'Embarked']

numeric\_transformer = Pipeline(steps=[ ('imputer', SimpleImputer(strategy='median')), ('scaler', StandardScaler())

])

categorical\_transformer = Pipeline(steps=[ ('imputer', SimpleImputer(strategy='constant',

fill\_value='missing')),

('onehot', OneHotEncoder(handle\_unknown='ignore'))

])

preprocessor = ColumnTransformer( transformers=[

('num', numeric\_transformer, numeric\_features), ('cat', categorical\_transformer, categorical\_features)

])

*# Split the data*

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

*# Create a pipeline that preprocesses the data then performs logistic regression*

model = Pipeline(steps=[('preprocessor', preprocessor),

('classifier', LogisticRegression(max\_iter=1000))])

*# Fit the model*

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(f'Model Accuracy: {accuracy:.2f}')

*# Generate a confusion matrix*

cm = confusion\_matrix(y\_test, y\_pred) plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues') plt.title('Confusion Matrix')

plt.ylabel('Actual') plt.xlabel('Predicted') plt.show()

*# Generate a classification report*

print(classification\_report(y\_test, y\_pred))

*# Interpret the Results*

*# Get feature names after preprocessing*

feature\_names = (model.named\_steps['preprocessor']

.named\_transformers\_['cat']

.named\_steps['onehot']

.get\_feature\_names\_out(categorical\_features)) feature\_names = np.concatenate([numeric\_features, feature\_names])

*# Get model coefficients*

coefficients = model.named\_steps['classifier'].coef\_[0]

*# Create a dataframe of features and their corresponding coefficients* feature\_importance = pd.DataFrame({'feature': feature\_names, 'importance': abs(coefficients)})

feature\_importance = feature\_importance.sort\_values('importance', ascending=False)

*# Plot feature importance*

plt.figure(figsize=(10, 6))

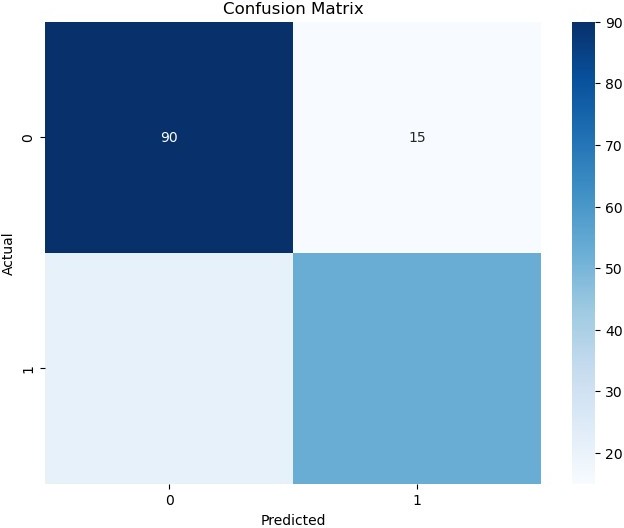
sns.barplot(x='importance', y='feature', data=feature\_importance) plt.title('Feature Importance')

precision recall f1-score support

plt.show()

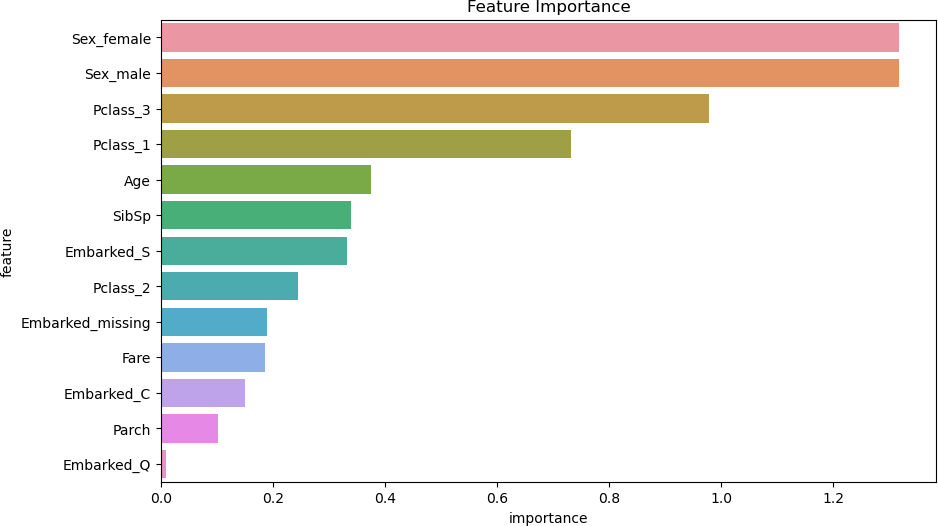
*# Print top 5 most important features* print("Top 5 most important features:") print(feature\_importance.head())

Model Accuracy: 0.80



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0.81 | 0.86 | 0.83 | 105 |
| 1 | 0.78 | 0.72 | 0.75 | 74 |
|  |  |  |  |  |
| accuracy |  |  | 0.80 | 179 |
| macro avg | 0.80 | 0.79 | 0.79 | 179 |
| weighted avg | 0.80 | 0.80 | 0.80 | 179 |

Top 5 most important features: feature importance



|  |  |  |
| --- | --- | --- |
| 7 | Sex\_female | 1.317635 |
| 8 | Sex\_male | 1.317533 |
| 6 | Pclass\_3 | 0.977835 |
| 4 | Pclass\_1 | 0.732311 |
| 0 | Age | 0.374346 |