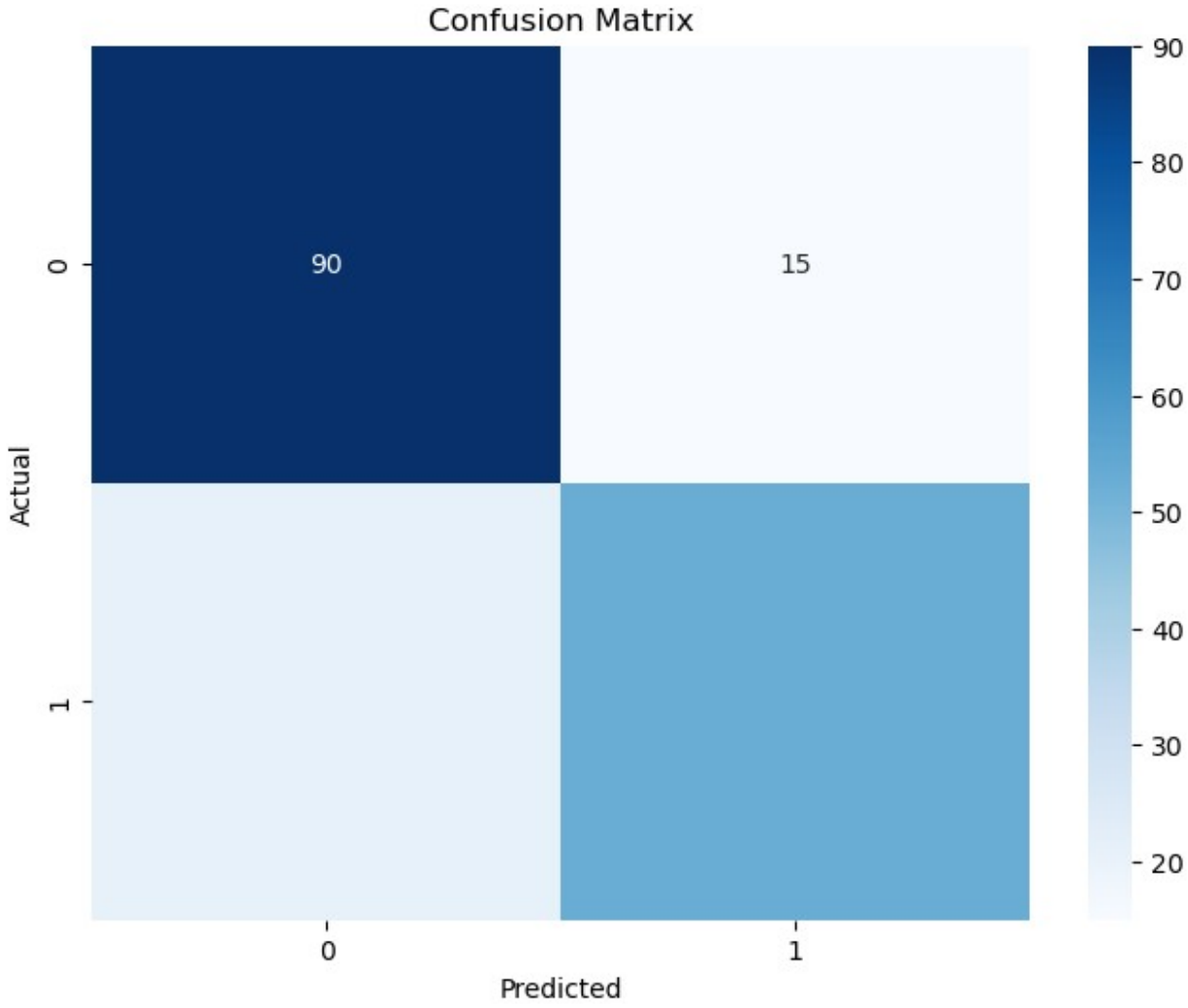
- Name: Deep Salunkhe
- Roll No.:21102A0014
- SEM-7 ML Lab2 Github Link

```
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)
```

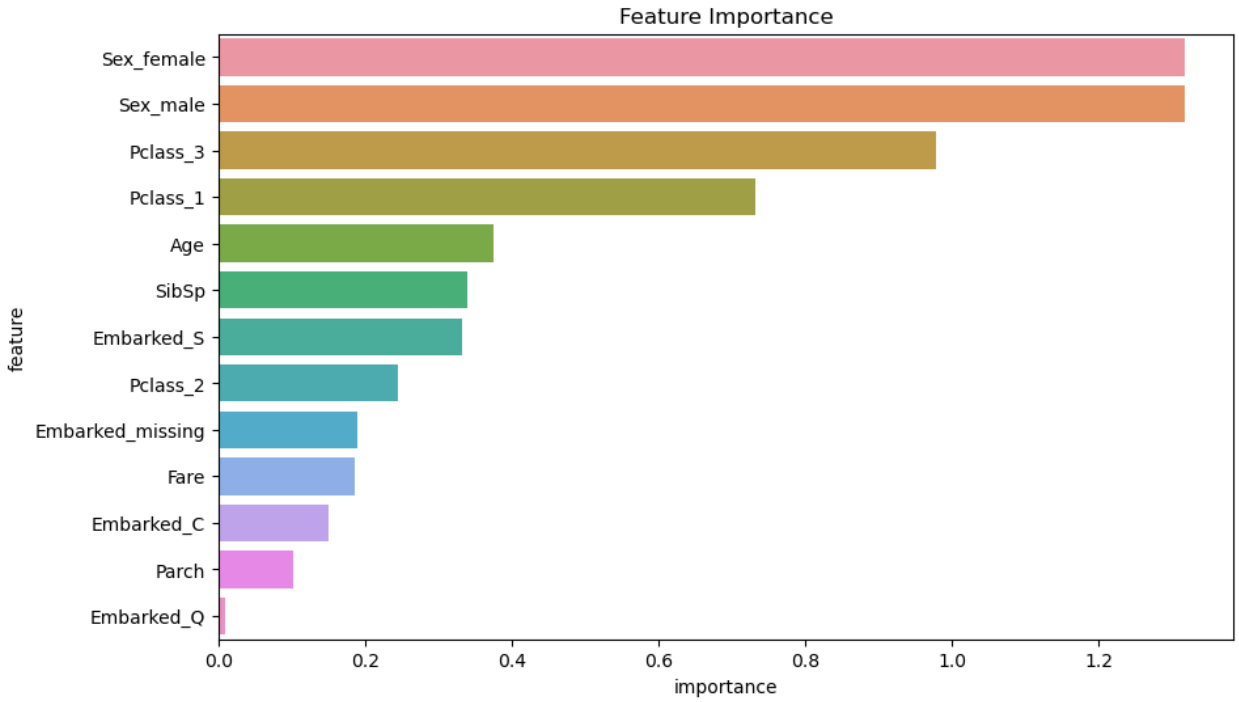
```
# Create a pipeline that preprocesses the data then performs logistic
regression
model = Pipeline(steps=[('preprocessor', preprocessor),
                        ('classifier',
LogisticRegression(max_iter=1000))])

# 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')
```

```
plt.show()

# Print top 5 most important features
print("Top 5 most important features:")
print(feature_importance.head())
```

```
Model Accuracy: 0.80
```




```
              precision    recall  f1-score   support

           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
```