In [48]:
```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
```

In [ ]:
```
```

In [49]:
```
# Load the Titanic dataset (assuming it's stored in a CSV file named 'titanic.csv')
url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"
titanic_data = pd.read_csv(url)
```

In [50]:
```
# Data Preprocessing
# Handle missing values
titanic_data = titanic_data.drop(['Cabin', 'Ticket', 'Name', 'PassengerId'], axis=1)  # I
titanic_data['Age'].fillna(titanic_data['Age'].median(), inplace=True)
titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0], inplace=True)
```

In [51]:
```
# Encode categorical variables
categorical_features = ['Sex', 'Embarked']
numeric_features = ['Pclass', 'Age', 'SibSp', 'Parch', 'Fare']
```

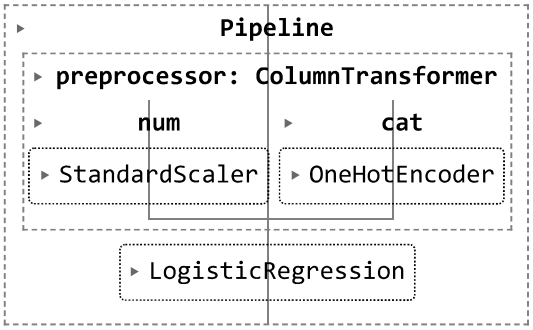
In [52]:
```
 #Create a column transformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])
```

In [53]:
```
# Define the model
model = Pipeline(steps=[('preprocessor', preprocessor),
                        ('classifier', LogisticRegression())])
```

In [54]:
```
# Train-Test Split
X = titanic_data.drop('Survived', axis=1)
y = titanic_data['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
```

In [55]:
```
# Model Training
model.fit(X_train, y_train)
```

Out[55]:



In [56]:
```
# Model Evaluation
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
conf_matrix = confusion_matrix(y_test, predictions)
classification_rep = classification_report(y_test, predictions)
```

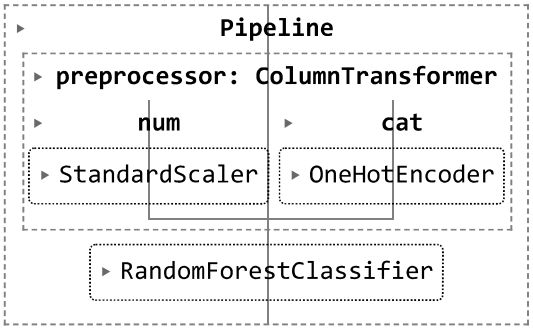
In [57]:
```
# Display Results
print(f'Accuracy: {accuracy}')
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Classification Report:\n{classification_rep}')
```

```
Accuracy: 0.8100558659217877
Confusion Matrix:
[[90 15]
 [19 55]]
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.86      0.84       105
           1       0.79      0.74      0.76        74

    accuracy                           0.81       179
   macro avg       0.81      0.80      0.80       179
weighted avg       0.81      0.81      0.81       179
```

In [58]:
```
from sklearn.ensemble import RandomForestClassifier

# Define a Random Forest model
rf_model = Pipeline(steps=[('preprocessor', preprocessor),
                           ('classifier', RandomForestClassifier(random_state=42))])
```

In [59]:
```
# Train Random Forest model
rf_model.fit(X_train, y_train)
```

Out[59]:



In [60]:
```
# Evaluate Random Forest model
rf_predictions = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)
rf_conf_matrix = confusion_matrix(y_test, rf_predictions)
rf_classification_rep = classification_report(y_test, rf_predictions)
```

In [61]:
```
# Compare Results
print("\nLogistic Regression Results:")
print(f'Accuracy: {accuracy}')
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Classification Report:\n{classification_rep}')
```

```
Logistic Regression Results:
Accuracy: 0.8100558659217877
Confusion Matrix:
[[90 15]
 [19 55]]
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.86      0.84       105
           1       0.79      0.74      0.76        74

    accuracy                           0.81       179
   macro avg       0.81      0.80      0.80       179
weighted avg       0.81      0.81      0.81       179
```

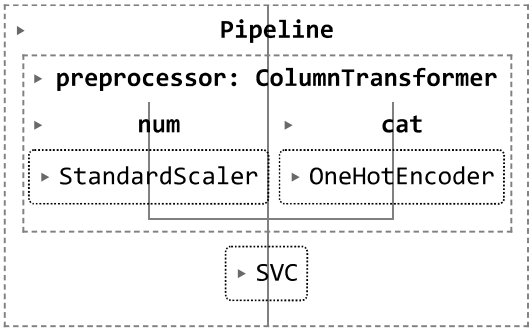
In [62]:
```
print("\nRandom Forest Results:")
print(f'Accuracy: {rf_accuracy}')
print(f'Confusion Matrix:\n{rf_conf_matrix}')
print(f'Classification Report:\n{rf_classification_rep}')
```

```
Random Forest Results:
Accuracy: 0.8044692737430168
Confusion Matrix:
[[89 16]
 [19 55]]
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.85      0.84       105
           1       0.77      0.74      0.76        74

    accuracy                           0.80       179
   macro avg       0.80      0.80      0.80       179
weighted avg       0.80      0.80      0.80       179
```

In [63]:
```
# Continue with the Support Vector Machine (SVM) section

# Define a Support Vector Machine model
svm_model = Pipeline(steps=[('preprocessor', preprocessor),
                            ('classifier', SVC(random_state=42))])
```

In [64]:
```
# Train Support Vector Machine model
svm_model.fit(X_train, y_train)
```

Out[64]:



In [65]:
```
# Evaluate Support Vector Machine model
svm_predictions = svm_model.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_predictions)
svm_conf_matrix = confusion_matrix(y_test, svm_predictions)
svm_classification_rep = classification_report(y_test, svm_predictions)
```

In [66]:
```
# Compare Results
print("\nSupport Vector Machine Results:")
print(f'Accuracy: {svm_accuracy}')
print(f'Confusion Matrix:\n{svm_conf_matrix}')
print(f'Classification Report:\n{svm_classification_rep}')
```

```
Support Vector Machine Results:
Accuracy: 0.8156424581005587
Confusion Matrix:
[[92 13]
 [20 54]]
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.88      0.85       105
           1       0.81      0.73      0.77        74

    accuracy                           0.82       179
   macro avg       0.81      0.80      0.81       179
weighted avg       0.82      0.82      0.81       179
```

In [67]:
```
# Identify the best model based on accuracy
best_model = max(["logistic regression", "random forest", "support vector machine"])
print(f"\nThe best model based on accuracy is: {best_model}")
```

```
The best model based on accuracy is: support vector machine
```

In [ ]:

In [ ]: