In [35]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error, r2_score
```

In [36]:

```
df = pd.read_csv('teams.csv')
```

In [37]:

```
# Preprocess the data
X = df.drop(["medals"], axis=1)  # Features
y = df["medals"]  # Target variable
```

In [38]:

```
categorical_features = ["team"]
numerical_features = ["year", "athletes", "events", "age", "height", "weight", "prev_m
```

In [50]:

```
categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])  # Set handle_unknown to "igr
numerical_transformer = Pipeline(steps=[('scale', 'passthrough')])
```

In [51]:

```
preprocessor = ColumnTransformer(transformers=[
    ('cat', categorical_transformer, categorical_features),
    ('num', numerical_transformer, numerical_features)])
```

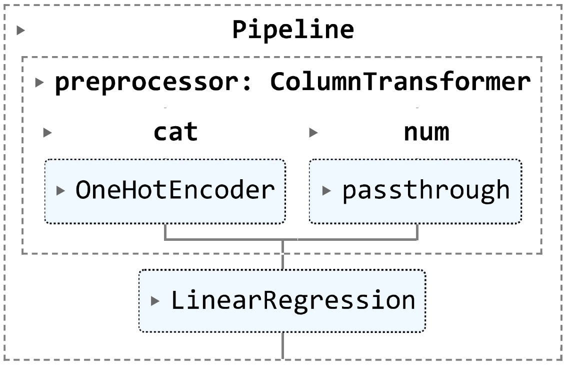
In [52]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [53]:

```
# Lienar Regression
lr_model = Pipeline(steps=[('preprocessor', preprocessor), ('regressor', LinearRegress
lr_model.fit(X_train, y_train)
```

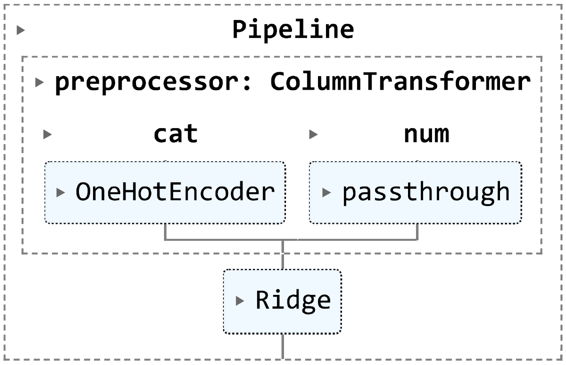
Out[53]:



In [54]:

```
# Ridge Regression
ridge_model = Pipeline(steps=[('preprocessor', preprocessor), ('regressor', Ridge(alph
ridge_model.fit(X_train, y_train)
```

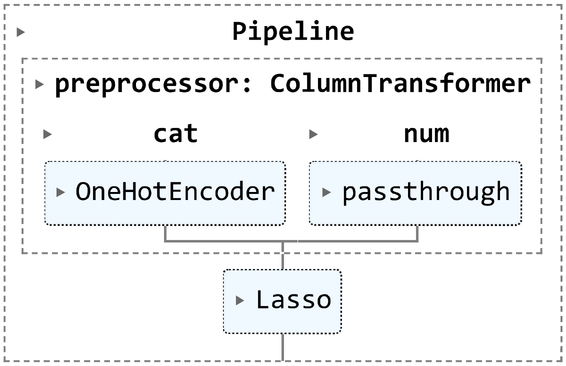
Out[54]:



In [55]:

```
# Lasso Regression
lasso_model = Pipeline(steps=[('preprocessor', preprocessor), ('regressor', Lasso(alph
lasso_model.fit(X_train, y_train)
```

Out[55]:



In [56]:

```
# Function to calculate RMSE and R-squared
def evaluate_model(model, X, y):
    y_pred = model.predict(X)
    mse = mean_squared_error(y, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y, y_pred)
    return rmse, r2
```

In [57]:

```
lr_rmse, lr_r2 = evaluate_model(lr_model, X_test, y_test)
print("Linear Regression:")
print("RMSE:", lr_rmse)
print("R-squared:", lr_r2)
```

```
Linear Regression:
RMSE: 10.643222590341596
R-squared: 0.847827593766797
```

In [58]:

```
ridge_rmse, ridge_r2 = evaluate_model(ridge_model, X_test, y_test)
print("\nRidge Regression:")
print("RMSE:", ridge_rmse)
print("R-squared:", ridge_r2)
```

```
Ridge Regression:
RMSE: 10.70925622928275
R-squared: 0.845933492700849
```

In [59]:

```
lasso_rmse, lasso_r2 = evaluate_model(lasso_model, X_test, y_test)
print("\nLasso Regression:")
print("RMSE:", lasso_rmse)
print("R-squared:", lasso_r2)
```

```
Lasso Regression:
RMSE: 10.930926316949385
R-squared: 0.8394894615529468
```

In [60]:
```
# Cross-validation
cross_val_rmse_lr = np.sqrt(-cross_val_score(lr_model, X, y, cv=5, scoring="neg_mean_s
cross_val_rmse_ridge = np.sqrt(-cross_val_score(ridge_model, X, y, cv=5, scoring="neg_
cross_val_rmse_lasso = np.sqrt(-cross_val_score(lasso_model, X, y, cv=5, scoring="neg_
```

In [61]:
```
print("\nCross-validation RMSE:")
print("Linear Regression:", cross_val_rmse_lr)
print("Ridge Regression:", cross_val_rmse_ridge)
print("Lasso Regression:", cross_val_rmse_lasso)
```

```
Cross-validation RMSE:
Linear Regression: 14.547464500210921
Ridge Regression: 13.081755132691667
Lasso Regression: 12.0888521487385
```

ANALYSIS

RMSE (Root Mean Squared Error): Lower RMSE values indicate better model performance. So, we want to choose the model with the lowest RMSE.

R-squared: R-squared measures how well the model fits the data. A higher R-squared value indicates a better fit. We want to choose the model with the highest R-squared.

Cross-Validation RMSE: Cross-validation provides an estimate of how the model will perform on unseen data. The model with the lowest cross-validation RMSE is likely to generalize better to new data.