In [15]:

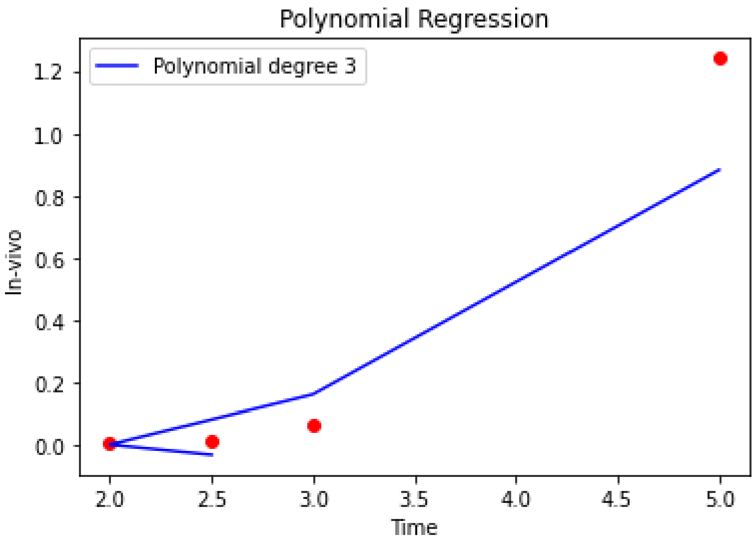
```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split

data = pd.read_csv('D:/class/Project/ML/sensitivity analysis/datasetcalc.csv')
#spliting
X = data[['time', 'in-vitro']].values
y = data['in-vivo'].values

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Create polynomial features for input data
#degree=5
poly = PolynomialFeatures(degree=5)
X_poly_train = poly.fit_transform(X_train)
X_poly_test = poly.transform(X_test)

# Fit polynomial regression model to training data
regressor = LinearRegression()
regressor.fit(X_poly_train, y_train)

# Use model to make predictions on test data
y_pred = regressor.predict(X_poly_test)

# Evaluate model performance using mean squared error and R-squared
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('Mean squared error: %.2f' % mse)
print('R-squared: %.2f' % r2)
```

```
Mean squared error: 0.02
R-squared: 0.91
```

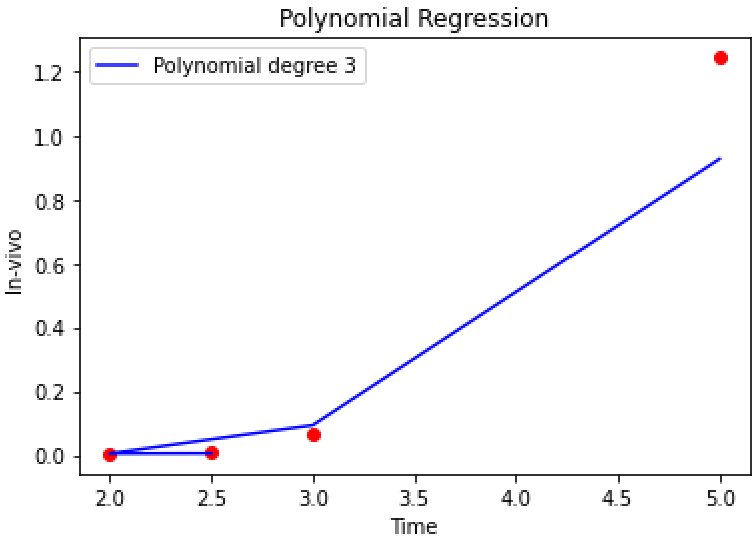
In [13]:
```
#for degree 3
poly = PolynomialFeatures(degree=3)
X_poly_train = poly.fit_transform(X_train)
X_poly_test = poly.transform(X_test)

# Fit polynomial regression model to training data
regressor = LinearRegression()
regressor.fit(X_poly_train, y_train)

# Use model to make predictions on test data
y_pred = regressor.predict(X_poly_test)

# Evaluate model performance using mean squared error and R-squared
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('Mean squared error: %.2f' % mse)
print('R-squared: %.2f' % r2)
```

```
Mean squared error: 0.03
R-squared: 0.87
```

In [9]:
```
#for degree 3
import matplotlib.pyplot as plt
poly3 = PolynomialFeatures(degree=3)
X_poly_train3 = poly3.fit_transform(X_train)
X_poly_test3 = poly3.transform(X_test)
regressor3 = LinearRegression()
regressor3.fit(X_poly_train3, y_train)
y_pred3 = regressor3.predict(X_poly_test3)
plt.scatter(X_test[:, 0], y_test, color='red')
plt.plot(X_test[:, 0], y_pred3, color='blue', label='Polynomial degree 3')
plt.title('Polynomial Regression')
plt.xlabel('Time')
plt.ylabel('In-vivo')
plt.legend()
plt.show()
```



In [12]:
```
import matplotlib.pyplot as plt
poly5 = PolynomialFeatures(degree=5)
X_poly_train5 = poly5.fit_transform(X_train)
X_poly_test5 = poly5.transform(X_test)
regressor5 = LinearRegression()
regressor5.fit(X_poly_train5, y_train)
y_pred5 = regressor5.predict(X_poly_test5)
plt.scatter(X_test[:, 0], y_test, color='red')
plt.plot(X_test[:, 0], y_pred5, color='blue', label='Polynomial degree 3')
plt.title('Polynomial Regression')
plt.xlabel('Time')
plt.ylabel('In-vivo')
plt.legend()
plt.show()
```



In [ ]:

In [ ]:

In [16]:
```
#test predict
import numpy as np

# Define arrays of time and invitro values
time_values = np.array([2,2.5,3,4,5,5.5])
invitro_values = np.array([0.31,0.42,0.53,0.79,0.9,0.92])

# Loop over combinations of time and invitro values and make predictions
for i in range(len(time_values)):
    # Create input array for current data point
    current_data = np.array([[time_values[i], invitro_values[i]]])

    # Create polynomial features for current data point
    current_data_poly = poly.transform(current_data)

    # Use model to make prediction for current data point
    current_invivo = regressor.predict(current_data_poly)

    # Print predicted invivo value for current data point
    print('Predicted invivo value for time =', time_values[i], 'and invitro =', invitro_values[i], ':', current_invivo
```

```
Predicted invivo value for time = 2.0 and invitro = 0.31 : [0.005172]
Predicted invivo value for time = 2.5 and invitro = 0.42 : [0.01165871]
Predicted invivo value for time = 3.0 and invitro = 0.53 : [0.06945036]
Predicted invivo value for time = 4.0 and invitro = 0.79 : [0.39983612]
Predicted invivo value for time = 5.0 and invitro = 0.9 : [0.92667373]
Predicted invivo value for time = 5.5 and invitro = 0.92 : [1.25393701]
```

In [20]:
```
#For RLD value
# Define arrays of time and invitro values
time_values = np.array([2,2.5,3,4,5,5.5])
invitro_values = np.array([0.23,0.38,0.5,0.76,0.83,0.9])

# Loop over combinations of time and invitro values and make predictions
for i in range(len(time_values)):
    # Create input array for current data point
    current_data = np.array([[time_values[i], invitro_values[i]]])

    # Create polynomial features for current data point
    current_data_poly = poly.transform(current_data)

    # Use model to make prediction for current data point
    current_invivo = regressor.predict(current_data_poly)

    # Print predicted invivo value for current data point
    print('Predicted invivo value for time =', time_values[i], 'and invitro =', invitro_values[i], ':', current_invivo
```

```
Predicted invivo value for time = 2.0 and invitro = 0.23 : [0.10270133]
Predicted invivo value for time = 2.5 and invitro = 0.38 : [0.05021096]
Predicted invivo value for time = 3.0 and invitro = 0.5 : [0.10947595]
Predicted invivo value for time = 4.0 and invitro = 0.76 : [0.451]
Predicted invivo value for time = 5.0 and invitro = 0.83 : [1.38980922]
Predicted invivo value for time = 5.5 and invitro = 0.9 : [1.46049919]
```

In [21]:

```
#For 5% lower value
# Define arrays of time and invitro values
time_values = np.array([2,2.5,3,4,5,5.5])
invitro_values = np.array([0.2,0.3,0.37,0.51,0.58,0.7])

# Loop over combinations of time and invitro values and make predictions
for i in range(len(time_values)):
    # Create input array for current data point
    current_data = np.array([[time_values[i], invitro_values[i]]])

    # Create polynomial features for current data point
    current_data_poly = poly.transform(current_data)

    # Use model to make prediction for current data point
    current_invivo = regressor.predict(current_data_poly)

    # Print predicted invivo value for current data point
    print('Predicted invivo value for time =', time_values[i], 'and invitro =', invitro_values[i], ':', current_invivo
```

```
Predicted invivo value for time = 2.0 and invitro = 0.2 : [0.17632081]
Predicted invivo value for time = 2.5 and invitro = 0.3 : [0.24902599]
Predicted invivo value for time = 3.0 and invitro = 0.37 : [0.59704699]
Predicted invivo value for time = 4.0 and invitro = 0.51 : [2.27015648]
Predicted invivo value for time = 5.0 and invitro = 0.58 : [6.93758861]
Predicted invivo value for time = 5.5 and invitro = 0.7 : [6.3935943]
```

In [ ]: