In [60]:
```python
import pandas as pd
import statsmodels.api as sm
```

1.) Import Data from FRED

In [61]:
```python
data = pd.read_csv("TaylorRuleData.csv", index_col = 0).dropna()
data.head()
```

Out[61]:

	FedFunds	Unemployment	HousingStarts	Inflation
1959-01-01	2.48	6.0	1657.0	29.01
1959-02-01	2.43	5.9	1667.0	29.00
1959-03-01	2.80	5.6	1620.0	28.97
1959-04-01	2.96	5.2	1590.0	28.98
1959-05-01	2.90	5.1	1498.0	29.04

In [62]:
```python
data.index = pd.to_datetime(data.index)
```

2.) Do Not Randomize, split your data into Train, Test Holdout

In [63]:
```python
split_1 = int(len(data)*0.6)
split_2 = int(len(data)*0.9)
data_in = data[:split_1]
data_out = data[split_1:split_2]
data_hold = data[split_2:]
```

In [64]:
```python
X_in = data_in.iloc[:,1:]
y_in = data_in.iloc[:,0]
X_out = data_out.iloc[:,1:]
y_out = data_out.iloc[:,0]
X_hold = data_hold.iloc[:,1:]
y_hold = data_hold.iloc[:,0]
```

In [65]:
```python
# Add Constants
X_in = sm.add_constant(X_in)
X_out = sm.add_constant(X_out)
X_hold = sm.add_constant(X_hold)
```

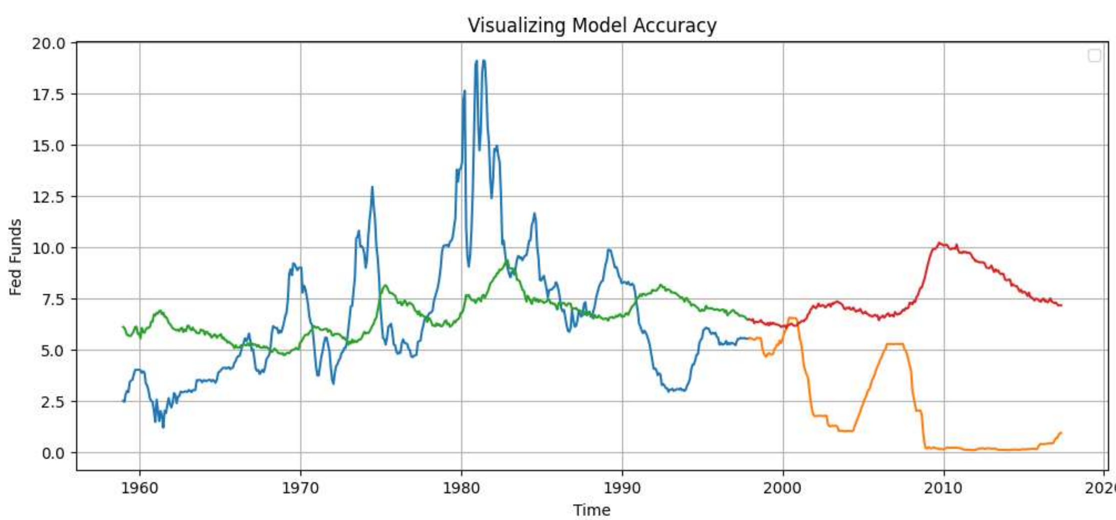
3.) Build a model that regresses FF~Unemp, HousingStarts, Inflation

In [66]:
```
model1 = sm.OLS(y_in, X_in).fit()
```

4.) Recreate the graph for your model

In [67]:
```
import matplotlib.pyplot as plt
```

In [68]:
```
plt.figure(figsize = (12,5))

###
plt.plot(y_in)
plt.plot(y_out)
plt.plot(model1.predict(X_in))
plt.plot(model1.predict(X_out))
###

plt.ylabel("Fed Funds")
plt.xlabel("Time")
plt.title("Visualizing Model Accuracy")
plt.legend([])
plt.grid()
plt.show()
```



"All Models are wrong but some are useful" - 1976 George Box

5.) What are the in/out of sample MSEs

In [69]:
```
from sklearn.metrics import mean_squared_error
```

In [70]:
```
in_mse_1 = mean_squared_error(model1.predict(X_in), y_in)
out_mse_1 = mean_squared_error(model1.predict(X_out), y_out)
```

In [71]:
```
print("Insample MSE : ", in_mse_1)
print("Outsample MSE : ", out_mse_1)
```

```
Insample MSE :  10.071422013168643
Outsample MSE :  40.36082783566727
```

6.) Using a for loop. Repeat 3,4,5 for polynomial degrees 1,2,3

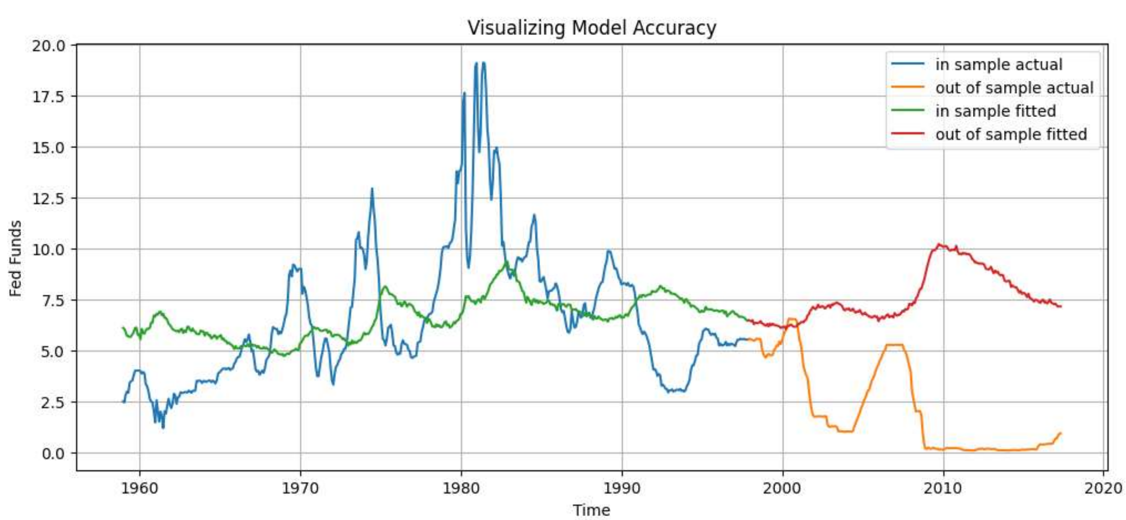
In [72]:
```
from sklearn.preprocessing import PolynomialFeatures
```

In [73]:
```
max_degrees = 3
```

In [75]:

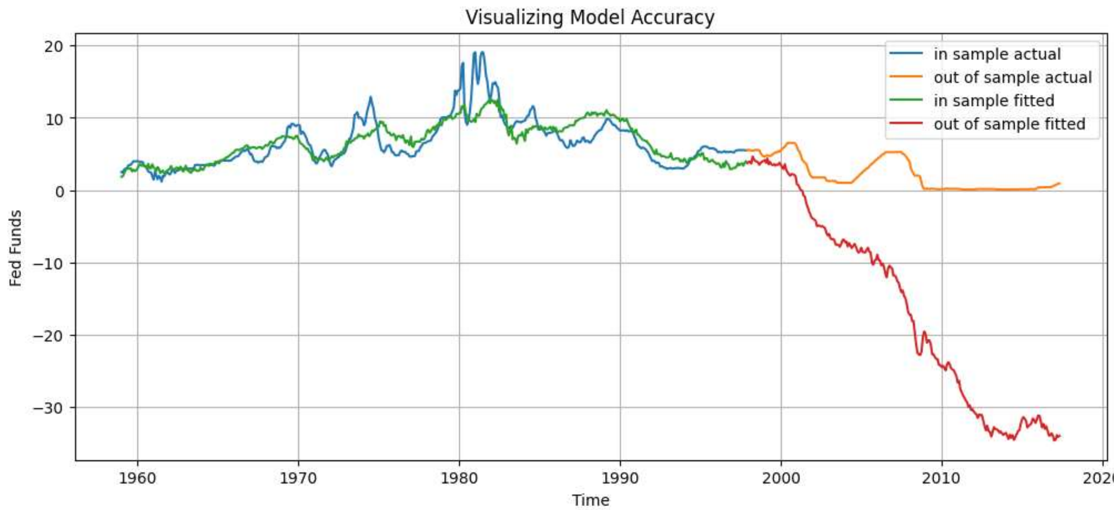
```python
for degrees in range(1, max_degrees+1):
    print("DEGREES : ", degrees)
    poly = PolynomialFeatures(degree = degrees)
    X_in_poly = poly.fit_transform(X_in)
    X_out_poly = poly.transform(X_out)

    #Q3
    model1 = sm.OLS(y_in, X_in_poly).fit()

    #Q4
    plt.figure(figsize = (12,5))

    in_preds = model1.predict(X_in_poly)
    in_preds = pd.DataFrame(in_preds, index = y_in.index)
    out_preds = model1.predict(X_out_poly)
    out_preds = pd.DataFrame(out_preds, index = y_out.index)

    plt.plot(y_in)
    plt.plot(y_out)
    plt.plot(in_preds)
    plt.plot(out_preds)

    plt.ylabel("Fed Funds")
    plt.xlabel("Time")
    plt.title("Visualizing Model Accuracy")
    plt.legend(["in sample actual", "out of sample actual", "in sample fit
    plt.grid()
    plt.show()

    #Q5
    in_mse_1 = mean_squared_error(model1.predict(X_in_poly), y_in)
    out_mse_1 = mean_squared_error(model1.predict(X_out_poly), y_out)
    print("Insample MSE : ", in_mse_1)
    print("Outsample MSE : ", out_mse_1)
    print("___________________________")
```

DEGREES : 1



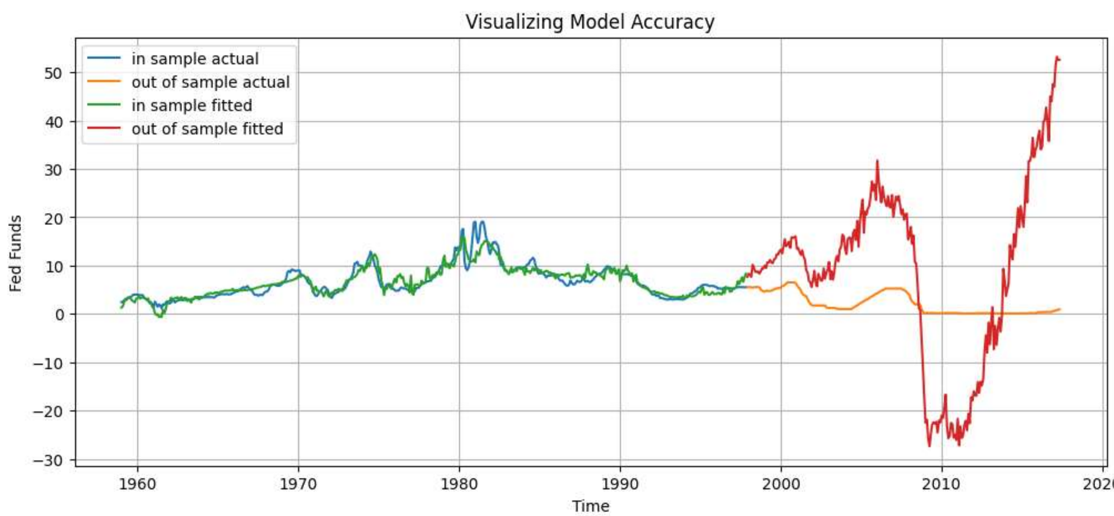
```
Insample MSE :  10.07142201316864
Outsample MSE :  40.36082783566782
```

```
DEGREES :  2
```




```
Insample MSE :  3.8634771392760685
Outsample MSE :  481.4465099294859
```

```
DEGREES :  3
```




```
Insample MSE :  1.8723636266506438
Outsample MSE :  371.7680409381023
```

7.) State your observations :

In the first graph, the model generally follows the in-sample data but does not capture the out-of-sample (test) data well. In the second graph, the model fits the in-sample data very well but performs the worst in fitting the out-of-sample data (highest out-of-sample MSE). In the third graph, the model fits the in-sample data the best (lowest in-sample MSE) but does not capture the out-of-sample data well.