Name: Geyu Zhang

UID: 502305399

In [1]:
```
import pandas as pd
import statsmodels.api as sm
```

1.) Import Data from FRED

In [2]:
```
data = pd.read_csv("TaylorRuleData.csv", index_col = 0)
```

In [3]:
```
data.index = pd.to_datetime(data.index)
```

In [4]:
```
data.describe()
```

Out[4]:

	FedFunds	Unemployment	HousingStarts	Inflation
count	834.000000	912.000000	779.000000	923.000000
mean	4.601667	5.704934	1433.207959	118.818295
std	3.592438	1.710877	382.765522	84.653938
min	0.050000	2.500000	478.000000	21.480000
25%	1.802500	4.400000	1204.000000	32.230000
50%	4.165000	5.500000	1457.000000	107.500000
75%	6.240000	6.700000	1649.500000	189.500000
max	19.100000	14.900000	2494.000000	307.917000

In [5]:
```
data.dropna(inplace = True)
```

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

In [7]:
```
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 [8]:
```
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 [9]:

```
# 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 [12]:

```
model1 = sm.OLS(y_in,X_in).fit()
```

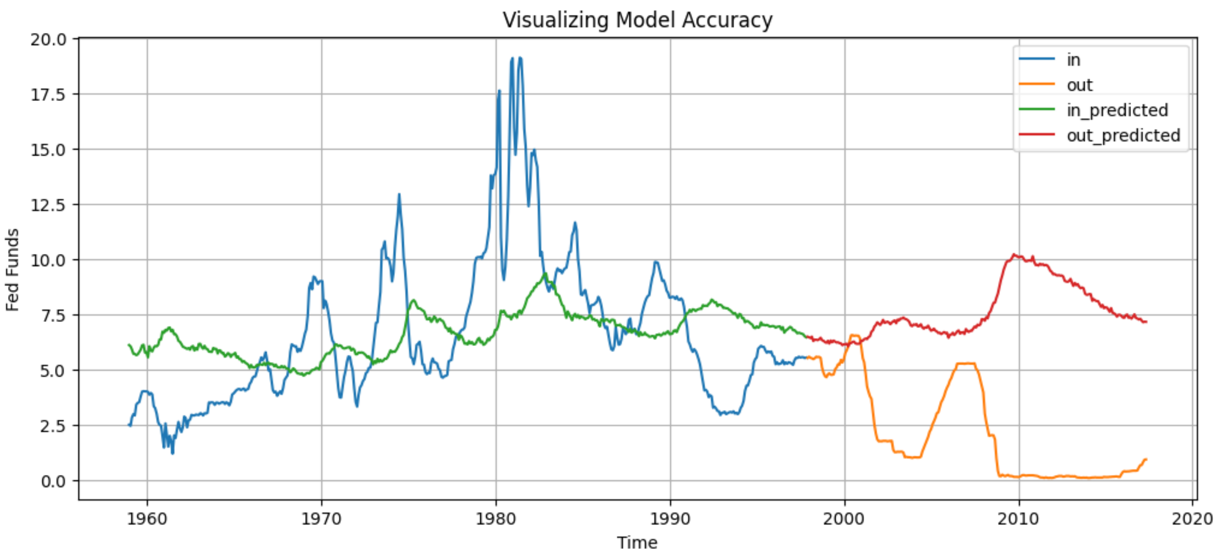
4.) Recreate the graph fro your model

In [13]:

```
import matplotlib.pyplot as plt
```

In [15]:

```
plt.figure(figsize = (12,5))

###

plt.plot(y_in, label = "in")
plt.plot(y_out, label = "out")
plt.plot(model1.predict(X_in), label = "in_predicted")
plt.plot(model1.predict(X_out), label = "out_predicted")

###

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 [16]:
```
from sklearn.metrics import mean_squared_error
```

In [17]:
```
in_mse_1 = mean_squared_error(y_in,model1.predict(X_in))
out_mse_1 = mean_squared_error(y_out, model1.predict(X_out))
```

In [18]:
```
print("Insample MSE : ", in_mse_1)
print("Outsample MSE : ", out_mse_1)
```

```
Insample MSE :  10.071422013168641
Outsample MSE :  40.360827835668566
```

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

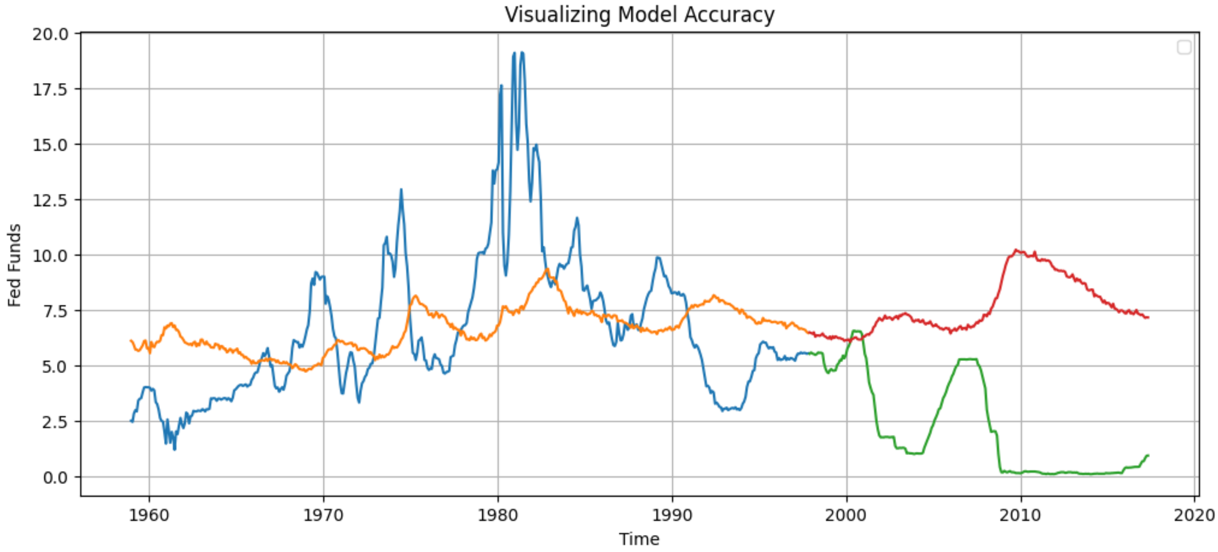
In [20]:
```
from sklearn.preprocessing import PolynomialFeatures
```

In [19]:
```
max_degrees = 3
```

In [21]:

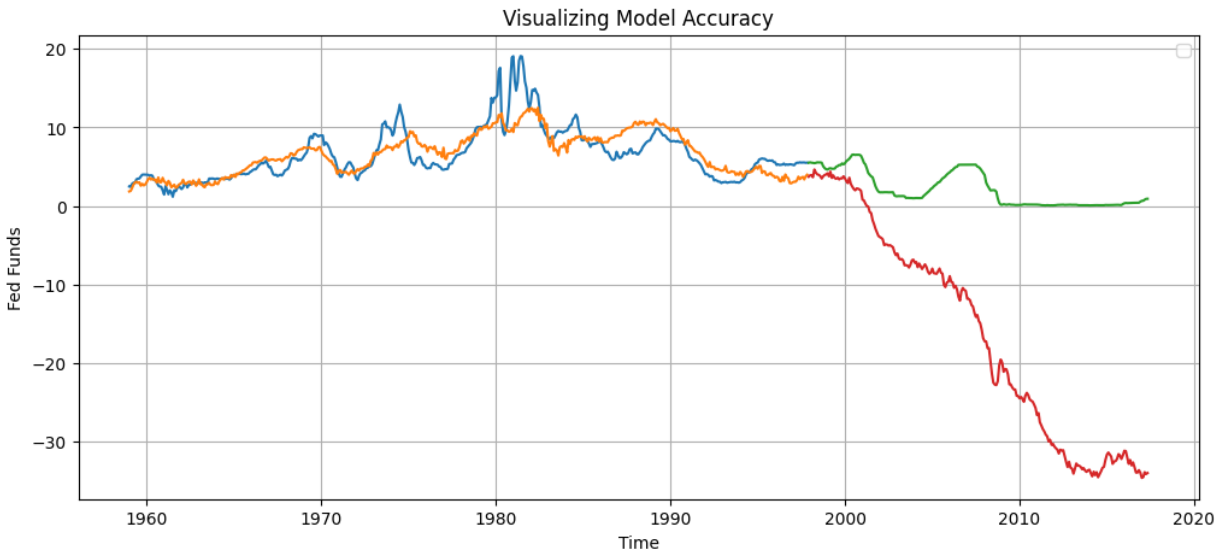
```
for degrees in range(1, max_degrees+1):
    print('DEGREE:', degrees)
    poly = PolynomialFeatures(degree = degrees)
    X_in_poly = poly.fit_transform(X_in)
    X_out_poly = poly.transform(X_out)    # without 'fit_'

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

    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)

    print(in_preds.shape)
    print(y_in.shape)

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

    ###

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

    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("In MSE:", in_mse_1)
    print("Out MSE:",out_mse_1)
    print('
```

```
DEGREE: 1
(467, 1)
(467,)
```



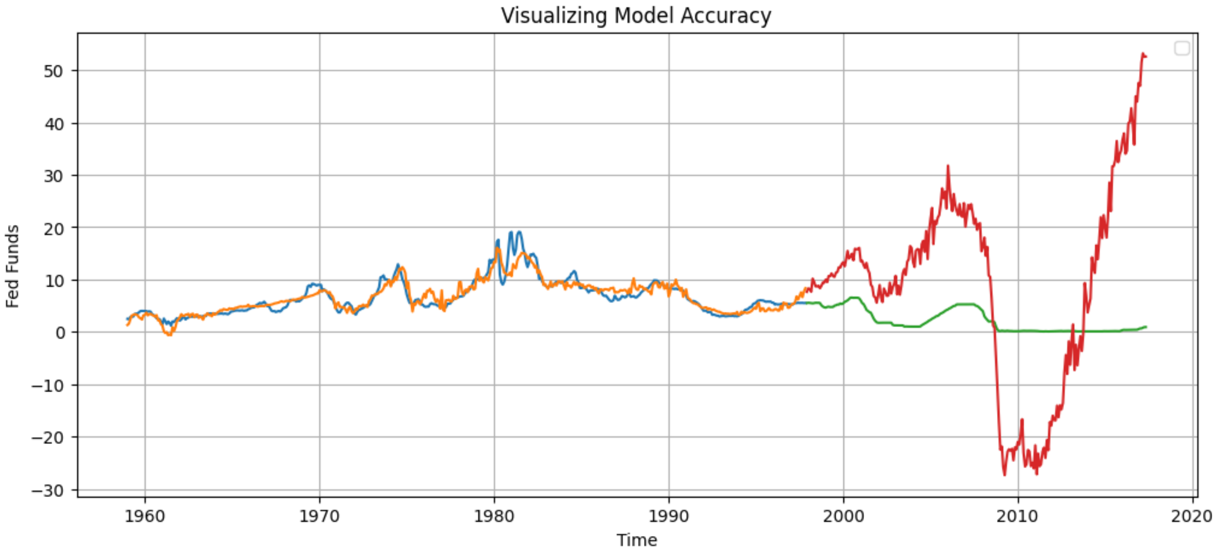

```
In MSE: 10.071422013168641
Out MSE: 40.36082783565212
```

```
DEGREE: 2
(467, 1)
(467,)
```




```
In MSE: 3.863477139276067
Out MSE: 481.4465099024015
```

```
DEGREE: 3
(467, 1)
(467,)
```




```
In MSE: 1.872363628831326
Out MSE: 371.7672613994206
```

7.) State your observations :

From the results, it's clear that as our model becomes more complex, it makes fewer mistakes on the training data (in-sample MSE decreases). But at the same time, it starts making more mistakes on new, unseen data (out-of-sample MSE increases). This means our model is becoming overfit, which is when it learns too much from the training data and can't perform well on new data. Essentially, our model is too focused on the details of the training set and fails to generalize to other data.