In [91]:
```
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

1.) Import Data from FRED ¶

In [58]:
```
data = pd.read_csv("TaylorRuleData.csv", index_col = 0)
data.head()
```

Out[58]:

	FedFunds	Unemployment	HousingStarts	Inflation
1947-01-01	NaN	NaN	NaN	21.48
1947-02-01	NaN	NaN	NaN	21.62
1947-03-01	NaN	NaN	NaN	22.00
1947-04-01	NaN	NaN	NaN	22.00
1947-05-01	NaN	NaN	NaN	21.95

In [59]:
```
data = data.dropna()
```

In [60]:
```
data.head()
```

Out[60]:

	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 [61]:
```
data.index = pd.to_datetime(data.index)
```

In [62]:
```
data.index
```

Out[62]:
```
DatetimeIndex(['1959-01-01', '1959-02-01', '1959-03-01', '1959-04-01',
               '1959-05-01', '1959-06-01', '1959-07-01', '1959-08-01',
               '1959-09-01', '1959-10-01',
               ...
               '2023-02-01', '2023-03-01', '2023-04-01', '2023-05-01',
               '2023-06-01', '2023-07-01', '2023-08-01', '2023-09-01',
               '2023-10-01', '2023-11-01'],
              dtype='datetime64[ns]', length=779, freq=None)
```

In [63]:
```
data.sample(len(data))
```

Out[63]:

	FedFunds	Unemployment	HousingStarts	Inflation
2015-11-01	0.12	5.1	1172.0	238.017
1974-05-01	11.31	5.1	1426.0	48.600
1992-05-01	3.82	7.6	1214.0	139.700
2011-12-01	0.07	8.5	694.0	227.223
1977-05-01	5.35	7.0	1971.0	60.200
...	...	...	...	...
1964-08-01	3.50	5.0	1569.0	31.050
1962-07-01	2.71	5.4	1450.0	30.220
1974-08-01	12.01	5.5	1142.0	49.900
2004-11-01	1.93	5.4	1782.0	191.700
2012-01-01	0.08	8.3	723.0	227.842

779 rows × 4 columns

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

In [64]:
```
split1 = int(len(data)*.6)
split2 = int(len(data)*.9)
data_in = data[:split1]
data_out = data[split1:split2]
data_hold = data[split2:]
```

In [65]:
```
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 [66]:
```
# Add Constants
X_in = sm.add_constant(X_in)
X_out = sm.add_constant(X_out)
X_hold =  sm.add_constant(X_hold)
```

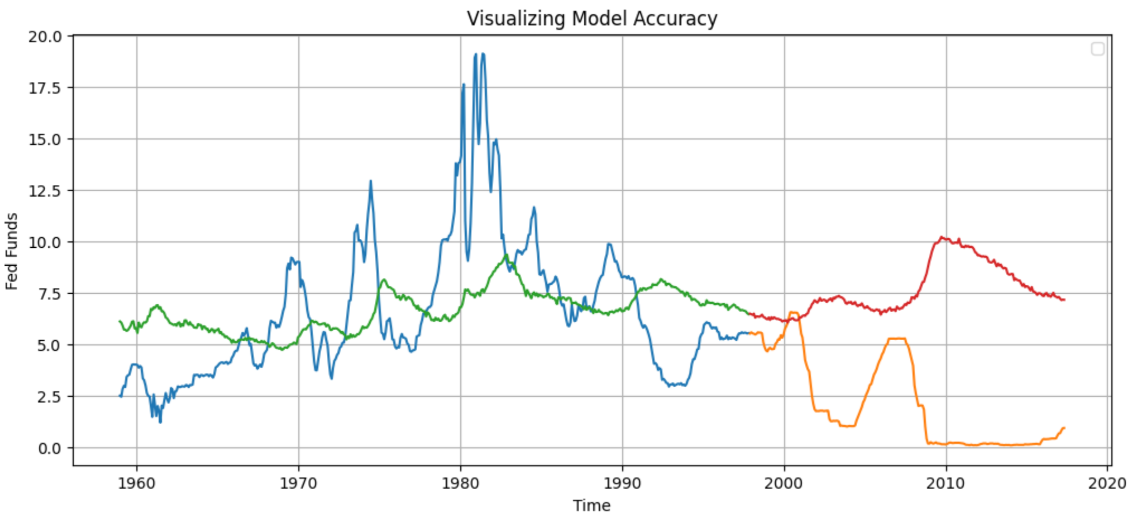
3.) Build a model that regresses FF~Unemp, HousingStarts, Inflation

In [92]:
```
model1 = sm.OLS(y_in, X_in).fit()
```

4.) Recreate the graph fro your model

In [93]:
```
import matplotlib.pyplot as plt
```

In [94]:
```
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 [95]:
```
from sklearn.metrics import mean_squared_error
```

In [96]:
```
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 [97]:
```
print("Insample MSE : ", in_mse_1)
print("Outsample MSE : ", out_mse_1)
```

```
Insample MSE :  10.071422013168641
Outsample MSE :  40.3608278356685
```

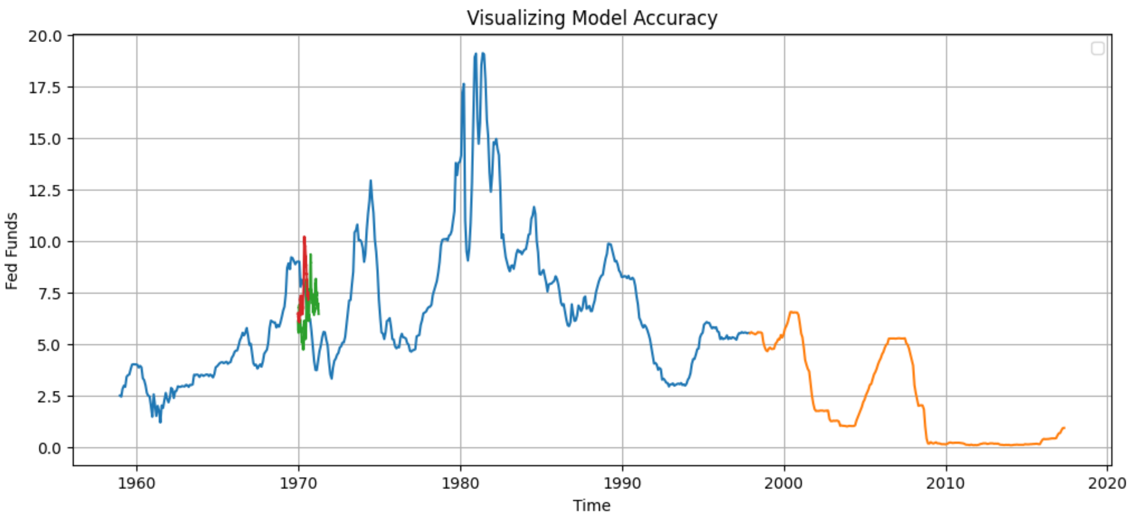
6.) Using a for loop. Repeat 3,4,5 for polynomial degrees 1,2,3

In [98]:
```
from sklearn.preprocessing import PolynomialFeatures
```

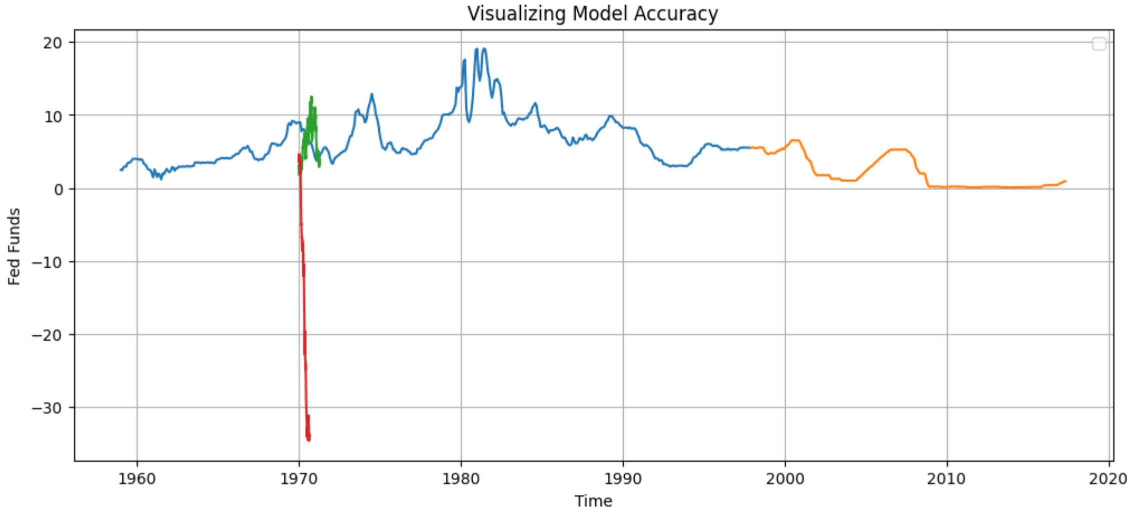
In [109]:
```
max_degrees = 3
```

In [110]:
```
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)

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

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

    pred_in = model1.predict(X_in_poly)
    pred_in = pd.DataFrame(pred_in, index = y_in.index)

    ###
    plt.plot(y_in)
    plt.plot(y_out)
    plt.plot(model1.predict(X_in_poly))
    plt.plot(model1.predict(X_out_poly))
    ###plt.plot(model1.predict(X_out))
    ###

    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("Insample MSE : ", in_mse_1)
    print("Outsample MSE : ", out_mse_1)
```

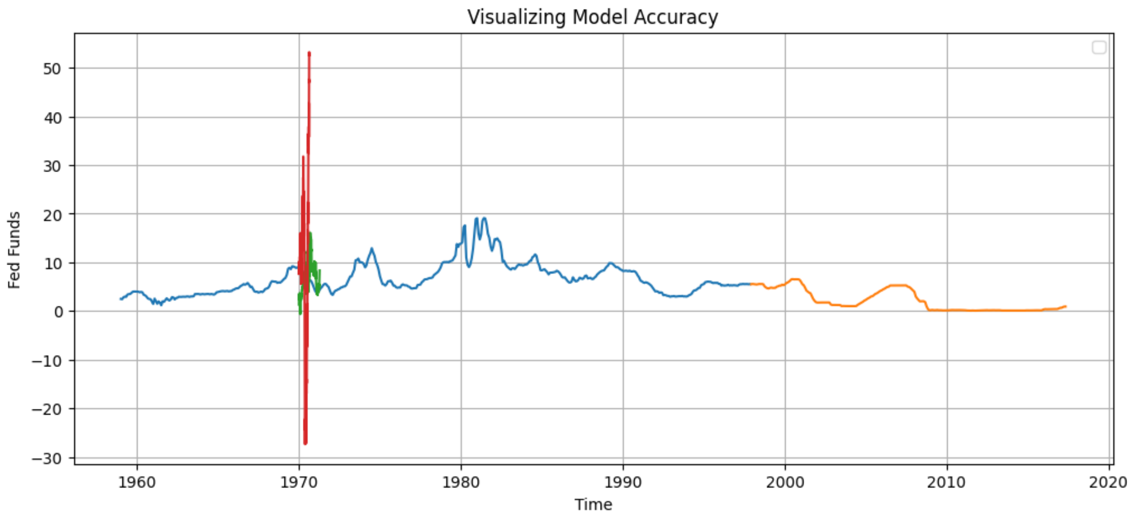
DEGREE: 1



Insample MSE : 10.071422013168641
Outsample MSE : 40.360827835666804
DEGREE: 2




```
Insample MSE :  3.863477139276068
Outsample MSE :  481.4465099024112
DEGREE: 3
```




```
Insample MSE :  1.8723636267986143
Outsample MSE :  371.7663885894949
```

7.) State your observations :

Type *Markdown* and LaTeX: α^2

As the degrees of the model increase the in sample predictions get better but out of sample prediction degrade or become worse

In [ ]: