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.dropna(inplace = True)
```

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

In [5]:
```
split1 = int(len(data) * .6)
split2 = int(len(data) * .9)
data_in = data[:split1]
data_out = data[split1:split2]
data_hold = data[split2:]
```

In [6]:
```
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 [7]:
```
# 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

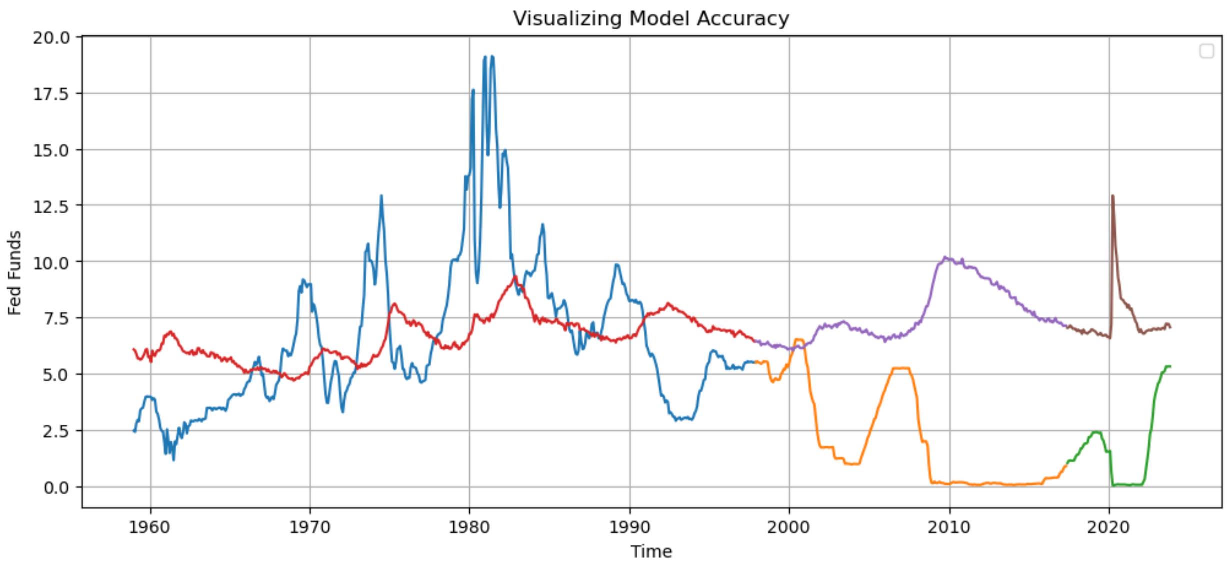
In [8]:
```
model1 = sm.OLS(y_in,X_in).fit()
```

4.) Recreate the graph fro your model

In [9]:
```
import matplotlib.pyplot as plt
```

In [10]:
```
plt.figure(figsize = (12,5))

###
```

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

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 [11]:
```
from sklearn.metrics import mean_squared_error
```

In [12]:
```
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 [13]:
```
print("Insample MSE : ", in_mse_1)
print("Outsample MSE : ", out_mse_1)
```

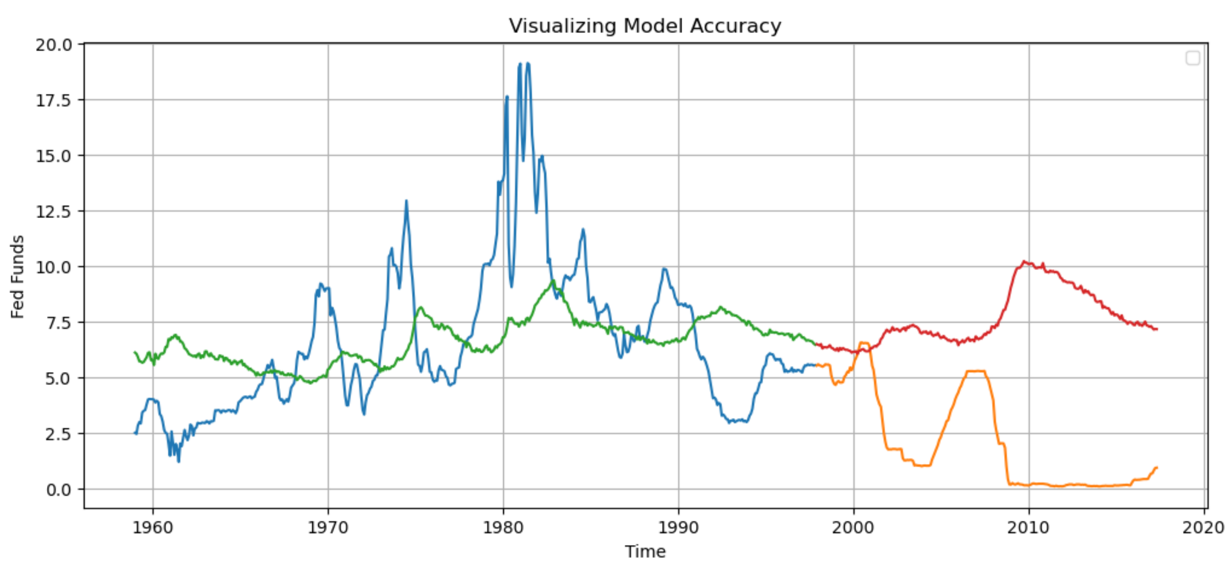
```
Insample MSE :  10.071422013168641
Outsample MSE :  40.36082783566723
```

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

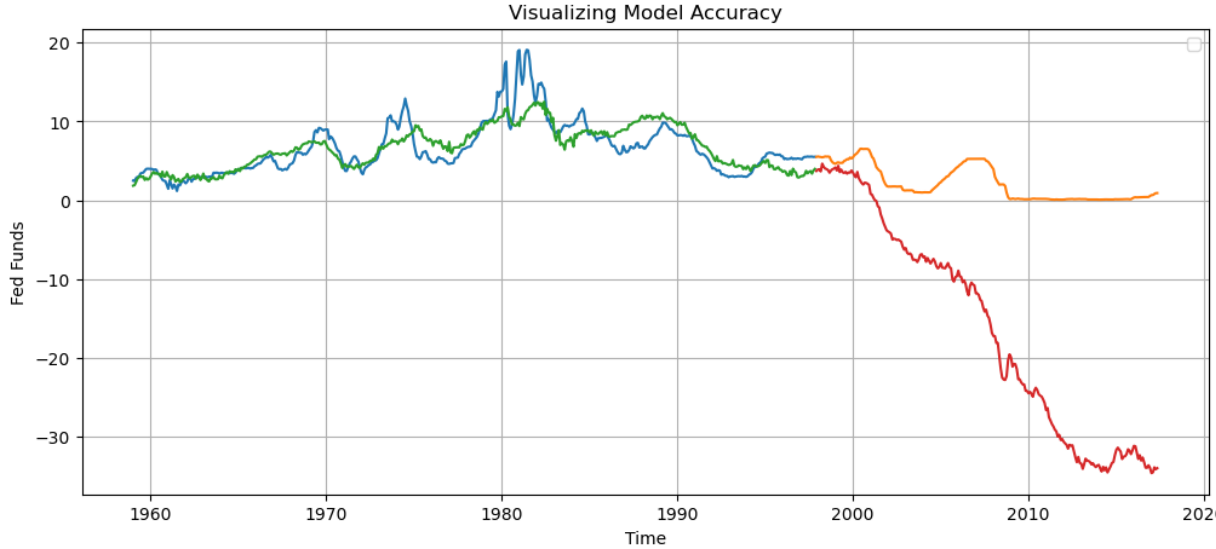
In [14]:
```
from sklearn.preprocessing import PolynomialFeatures
```

In [15]:
```
degrees = 1
```

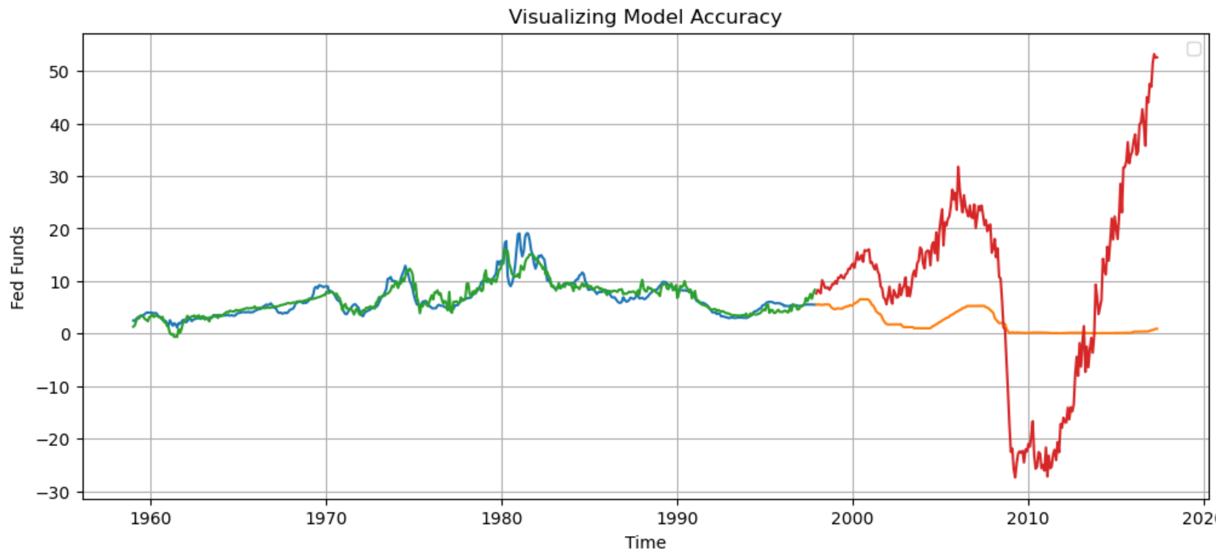
In [19]:
```
for degrees in range (1,4):
    poly = PolynomialFeatures(degree = degrees)
    X_in_poly = poly.fit_transform(X_in)
    X_out_poly = poly.fit_transform(X_out)


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

    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.figure(figsize = (12,5))

    ###
    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([])
    plt.grid()
    plt.show()
    in_mse_1 = mean_squared_error(y_in,model1.predict(X_in_poly))
    out_mse_1 = mean_squared_error(y_out,model1.predict(X_out_poly))
    print("Insample MSE : ", in_mse_1)
    print("Outsample MSE : ", out_mse_1)
```




```
Insample MSE :  10.071422013168641
Outsample MSE :  40.36082783565256
```




```
Insample MSE :  3.8634771392760685
Outsample MSE :  481.44650988981857
```




```
Insample MSE :  1.8723636291944272
Outsample MSE :  371.7674950018752
```

In [ ]:

```
poly = PolynomialFeatures(degree = degrees)
X_in_poly = poly.fit_transform(X_in)
X_out_poly = poly.fit_transform(X_out)
```

7.) State your observations :

First model has the most insample errors and appears to be underfit and performs poorly on both in and out of sample, indicating the sample may be underfit Second model performs better on insample data but performs very poorly for out of sample Third model similar to second model but to an ever greater extent, indicating that the second and third models may be overfit