Fed Funds Prediction

January 11, 2024

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

1 1.) Import Data from FRED

[2]:
```python
data = pd.read_csv("/Users/sandinatatu/Desktop/TaylorRuleData.csv", index_col =␣
↪0)
```

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

[4]:
```python
data.dropna(inplace=True)
```

[5]:
```python
data.head()
```

[5]:

	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

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

[6]:
```python
split_1 = int(len(data) * .6)
split_2 = int(len(data) * .9)
data_in = data[:split_1]
data_out = data[split_1:split_2]
data_hold = data[split_2:]
```

[7]:
```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]
```

[8]:
```python
# Add Constants
X_in = sm.add_constant(X_in)
X_out =  sm.add_constant(X_out)
X_hold =  sm.add_constant(X_hold)
```

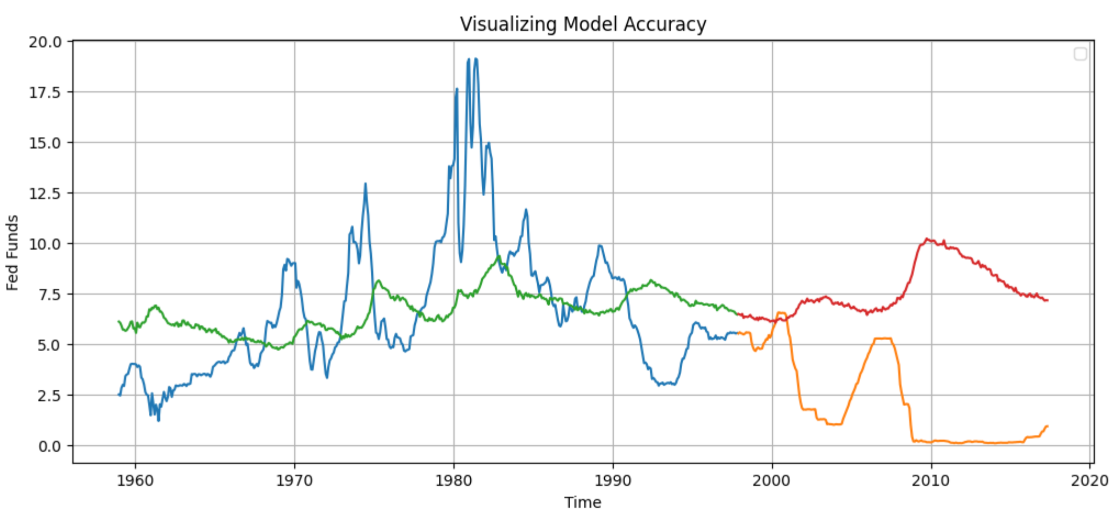
3 3.) Build a model that regresses FF~Unemp, HousingStarts, Inflation

[9]:
```python
model1 = sm.OLS(y_in, X_in).fit()
```

4 4.) Recreate the graph fro your model

[10]:
```python
import matplotlib.pyplot as plt
```

[11]:
```python
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()
```



4.1 “All Models are wrong but some are useful” - 1976 George Box

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

[14]:
```
from sklearn.metrics import mean_squared_error
```

[15]:
```
in_mse_1 = mean_squared_error(y_in, model1.predict(X_in))
out_mse_1 = mean_squared_error(y_out, model1.predict(X_out))
```

[16]:
```
print("Insample MSE : ", in_mse_1)
print("Outsample MSE : ", out_mse_1)
```

```
Insample MSE :  10.071422013168643
Outsample MSE :  40.3608278356685
```

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

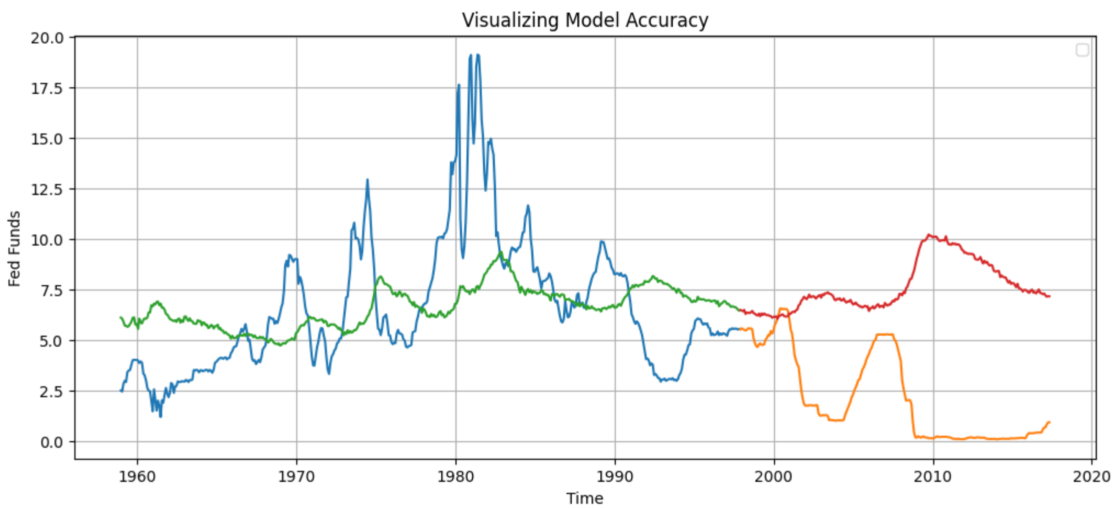
[20]:
```
from sklearn.preprocessing import PolynomialFeatures
```

[21]:
```
max_degrees = 3
```

[26]:
```
for degrees in range(1,1+max_degrees):
    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()

    #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([])
    plt.grid()
    plt.show()

    #Q5.)
    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)
    print("______________________________________")
    print("______________________________________")
```

DEGREES : 1

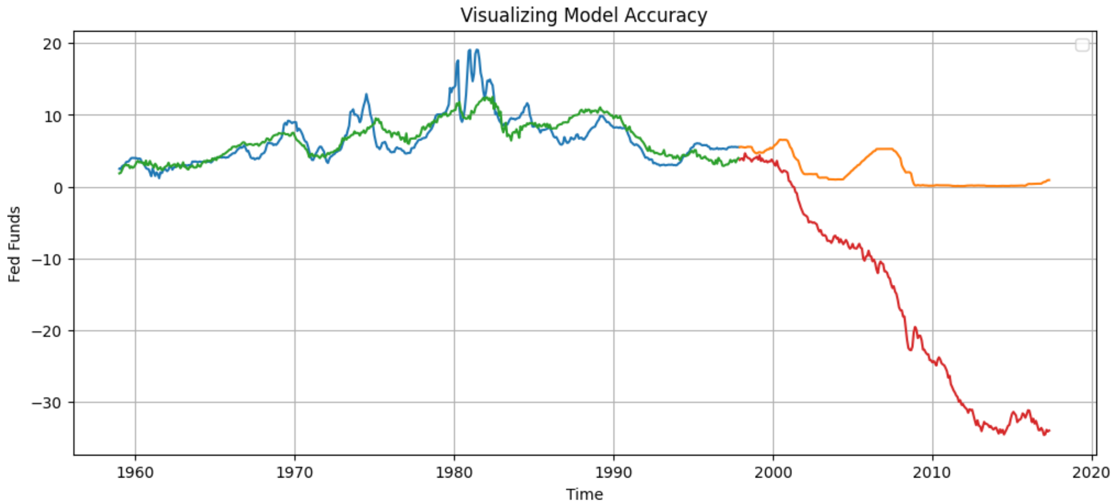


Insample MSE : 10.071422013168641
Outsample MSE : 40.36082783565204

____

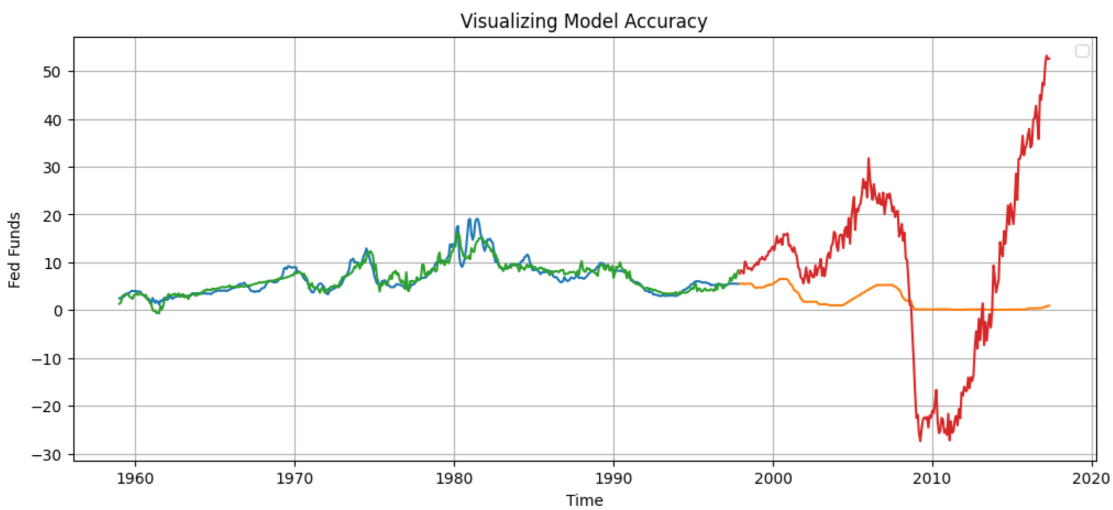
____

DEGREES : 2



Insample MSE : 3.863477139276068
Outsample MSE : 481.4465099024405

DEGREES : 3



Insample MSE : 1.8723636288250916
Outsample MSE : 371.7672642959744

----

7 7.) State your observations :

The Out-Of-Sample MSE is lowest for the model with one degrees. The other two models significantly overfit the data, leading to a much higher out-of-sample MSE.