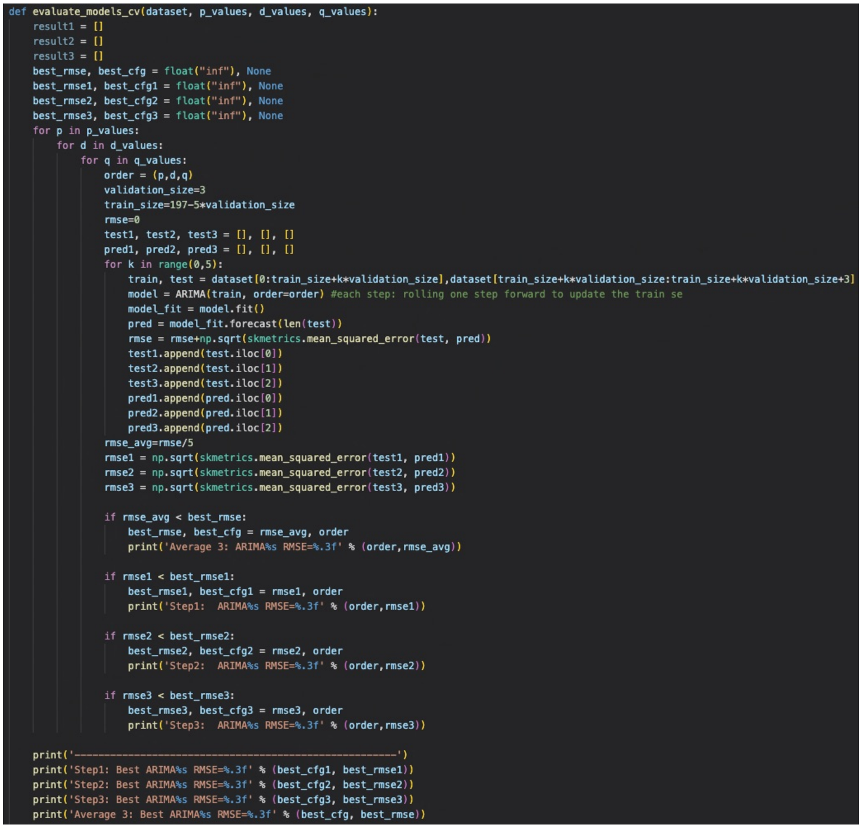
1.

method 1:

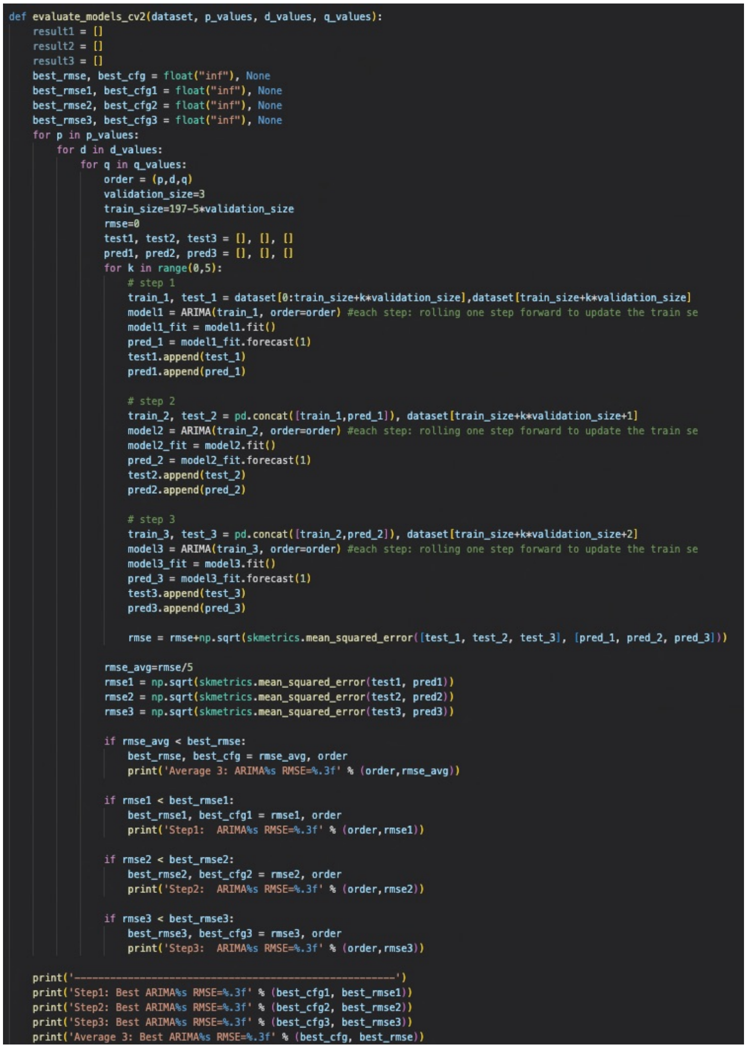
	Step 1	Step 2	Step 3	Avg 3 step
Best RMSE	0.854	0.699	0.735	0.793
Best model	(0, 0)	(0, 0)	(0, 4)	(0.0)



```python
def evaluate_models_cv(dataset, p_values, d_values, q_values):
    result1 = []
    result2 = []
    result3 = []
    best_rmse, best_cfg = float("inf"), None
    best_rmse1, best_cfg1 = float("inf"), None
    best_rmse2, best_cfg2 = float("inf"), None
    best_rmse3, best_cfg3 = float("inf"), None
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p,d,q)
                validation_size=3
                train_size=197-5*validation_size
                rmse=0
                test1, test2, test3 = [], [], []
                pred1, pred2, pred3 = [], [], []
                for k in range(0,5):
                    train, test = dataset[0:train_size+k*validation_size],dataset[train_size+k*validation_size:train_size+k*validation_size+3]
                    model = ARIMA(train, order=order) #each step: rolling one step forward to update the train se
                    model_fit = model.fit()
                    pred = model_fit.forecast(len(test))
                    rmse = rmse+np.sqrt(skmetrics.mean_squared_error(test, pred))
                    test1.append(test.iloc[0])
                    test2.append(test.iloc[1])
                    test3.append(test.iloc[2])
                    pred1.append(pred.iloc[0])
                    pred2.append(pred.iloc[1])
                    pred3.append(pred.iloc[2])
                rmse_avg=rmse/5
                rmse1 = np.sqrt(skmetrics.mean_squared_error(test1, pred1))
                rmse2 = np.sqrt(skmetrics.mean_squared_error(test2, pred2))
                rmse3 = np.sqrt(skmetrics.mean_squared_error(test3, pred3))

                if rmse_avg < best_rmse:
                    best_rmse, best_cfg = rmse_avg, order
                    print('Average 3: ARIMA%s RMSE=%.3f' % (order,rmse_avg))

                if rmse1 < best_rmse1:
                    best_rmse1, best_cfg1 = rmse1, order
                    print('Step1:  ARIMA%s RMSE=%.3f' % (order,rmse1))

                if rmse2 < best_rmse2:
                    best_rmse2, best_cfg2 = rmse2, order
                    print('Step2:  ARIMA%s RMSE=%.3f' % (order,rmse2))

                if rmse3 < best_rmse3:
                    best_rmse3, best_cfg3 = rmse3, order
                    print('Step3:  ARIMA%s RMSE=%.3f' % (order,rmse3))

    print('--------------------------------------------------------')
    print('Step1: Best ARIMA%s RMSE=%.3f' % (best_cfg1, best_rmse1))
    print('Step2: Best ARIMA%s RMSE=%.3f' % (best_cfg2, best_rmse2))
    print('Step3: Best ARIMA%s RMSE=%.3f' % (best_cfg3, best_rmse3))
    print('Average 3: Best ARIMA%s RMSE=%.3f' % (best_cfg, best_rmse))
```

2. method 2:

	Step1	Step2	Step3	Avg 3 step
Best RMSE	0,854	0.699	0,735	0.793
Best model	(0,0)	(0.0)	(0.4)	(0,0)



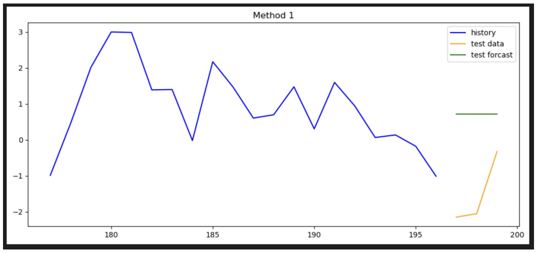
```
def evaluate_models_cv2(dataset, p_values, d_values, q_values):
    result1 = []
    result2 = []
    result3 = []
    best_rmse, best_cfg = float("inf"), None
    best_rmse1, best_cfg1 = float("inf"), None
    best_rmse2, best_cfg2 = float("inf"), None
    best_rmse3, best_cfg3 = float("inf"), None
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p,d,q)
                validation_size=3
                train_size=197-5*validation_size
                rmse=0
                test1, test2, test3 = [], [], []
                pred1, pred2, pred3 = [], [], []
                for k in range(0,5):
                    # step 1
                    train_1, test_1 = dataset[0:train_size+k*validation_size],dataset[train_size+k*validation_size]
                    model1 = ARIMA(train_1, order=order) #each step: rolling one step forward to update the train se
                    model1_fit = model1.fit()
                    pred_1 = model1_fit.forecast(1)
                    test1.append(test_1)
                    pred1.append(pred_1)

                    # step 2
                    train_2, test_2 = pd.concat([train_1,pred_1]), dataset[train_size+k*validation_size+1]
                    model2 = ARIMA(train_2, order=order) #each step: rolling one step forward to update the train se
                    model2_fit = model2.fit()
                    pred_2 = model2_fit.forecast(1)
                    test2.append(test_2)
                    pred2.append(pred_2)

                    # step 3
                    train_3, test_3 = pd.concat([train_2,pred_2]), dataset[train_size+k*validation_size+2]
                    model3 = ARIMA(train_3, order=order) #each step: rolling one step forward to update the train se
                    model3_fit = model3.fit()
                    pred_3 = model3_fit.forecast(1)
                    test3.append(test_3)
                    pred3.append(pred_3)

                    rmse = rmse+np.sqrt(skmetrics.mean_squared_error([test_1, test_2, test_3], [pred_1, pred_2, pred_3]))

                rmse_avg=rmse/5
                rmse1 = np.sqrt(skmetrics.mean_squared_error(test1, pred1))
                rmse2 = np.sqrt(skmetrics.mean_squared_error(test2, pred2))
                rmse3 = np.sqrt(skmetrics.mean_squared_error(test3, pred3))

                if rmse_avg < best_rmse:
                    best_rmse, best_cfg = rmse_avg, order
                    print('Average 3: ARIMA%s RMSE=%.3f' % (order,rmse_avg))

                if rmse1 < best_rmse1:
                    best_rmse1, best_cfg1 = rmse1, order
                    print('Step1:  ARIMA%s RMSE=%.3f' % (order,rmse1))

                if rmse2 < best_rmse2:
                    best_rmse2, best_cfg2 = rmse2, order
                    print('Step2:  ARIMA%s RMSE=%.3f' % (order,rmse2))

                if rmse3 < best_rmse3:
                    best_rmse3, best_cfg3 = rmse3, order
                    print('Step3:  ARIMA%s RMSE=%.3f' % (order,rmse3))

    print('----------------------------------------------------')
    print('Step1: Best ARIMA%s RMSE=%.3f' % (best_cfg1, best_rmse1))
    print('Step2: Best ARIMA%s RMSE=%.3f' % (best_cfg2, best_rmse2))
    print('Step3: Best ARIMA%s RMSE=%.3f' % (best_cfg3, best_rmse3))
    print('Average 3: Best ARIMA%s RMSE=%.3f' % (best_cfg, best_rmse))
```

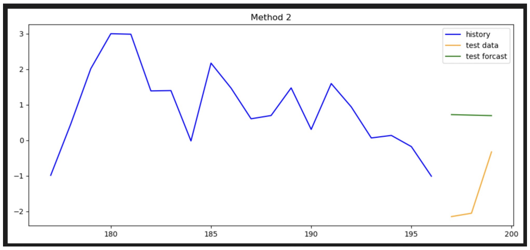
3. Method1 – RMSE: 2,38284449

Method2 – RMSE: 2,38284447

method2 is better (slightly)

4.


Method 1
history
test data
test forcast


Method 2
history
test data
test forcast