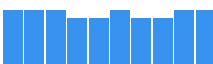




```
import pandas as pd
time_series = pd.read_csv('time_series.csv')
# drop_na = pd.read_csv('drop_na.csv')
```

time_series

	geo_value int64 6001 - 6115 	time_value object 2020-03-02 0.3% 2020-03-03 0.3% 362 others 99.5%	change_health fl... 0.0013741 - 12.2... 	hospital_admit fl... 0.077083 - 31.66... 	doc_visits float64	grou
0	6001	2020-03-02	0.0078964	0.119067	0.0	
1	6003	2020-03-02	0.053798476363 6363	0.179150153846 1538	0.108915435897 4359	
2	6005	2020-03-02	0.0805153	0.179150153846 1538	0.0	
3	6007	2020-03-02	0.065189	0.179150153846 1538	0.0	
4	6009	2020-03-02	0.0978474	0.179150153846 1538	0.108915435897 4359	
5	6011	2020-03-02	0.0927644	0.179150153846 1538	0.0	
6	6013	2020-03-02	0.017263	0.123715	0.0	
7	6015	2020-03-02	0.0860585	0.179150153846 1538	0.108915435897 4359	
8	6017	2020-03-02	0.0364431	0.179150153846 1538	0.324427	
9	6019	2020-03-02	0.0079491	0.179150153846 1538	0.0	

```
import numpy as np

dates = time_series['time_value'].unique()
dates_length = len(dates)
test_split = int(0.95*dates_length)

train_row = dates[:test_split]
test_row = dates[test_split:]

train = time_series.loc[time_series['time_value'].isin(train_row), :].set_index('time_value')
test = time_series.loc[time_series['time_value'].isin(test_row), :].set_index('time_value')
```

```

columns = time_series.columns
columns = np.append(columns, columns[6])
columns = np.delete(columns, [0,1,6])

X_train = train.loc[:, columns[:-1]]
y_train = train.loc[:, columns[-1]]
X_test = test.loc[:, columns[:-1]]
y_test = test.loc[:, columns[-1]]

```

Simple Model

```

from sklearn.model_selection import TimeSeriesSplit, cross_validate
from sklearn.tree import DecisionTreeRegressor
import numpy as np

tree_depths = range(2,17)
tscv = TimeSeriesSplit(n_splits=7)
train_avg = []
valid_avg = []
test_avg = []
for depth in tree_depths:
    dtree = DecisionTreeRegressor(max_depth=depth, random_state=0)
    cv_score = cross_validate(dtree, X_train, y_train, cv=tscv, scoring='r2', return_train_score=True)
    train_avg.append(cv_score['train_score'].mean())
    valid_avg.append(cv_score['test_score'].mean())

    tests = []
    for i, (train_index, validate_index) in enumerate(tscv.split(X_train)):
        temp_X_train = X_train.iloc[train_index]
        temp_y_train = y_train.iloc[train_index]
        temp_X_valid = X_train.iloc[validate_index]
        temp_y_valid = y_train.iloc[validate_index]

        dtree = DecisionTreeRegressor(max_depth=depth, random_state=0)
        dtree.fit(temp_X_train, temp_y_train)
        tests.append(dtree.score(temp_X_valid, temp_y_valid))

    test_avg.append(np.mean(tests))

```

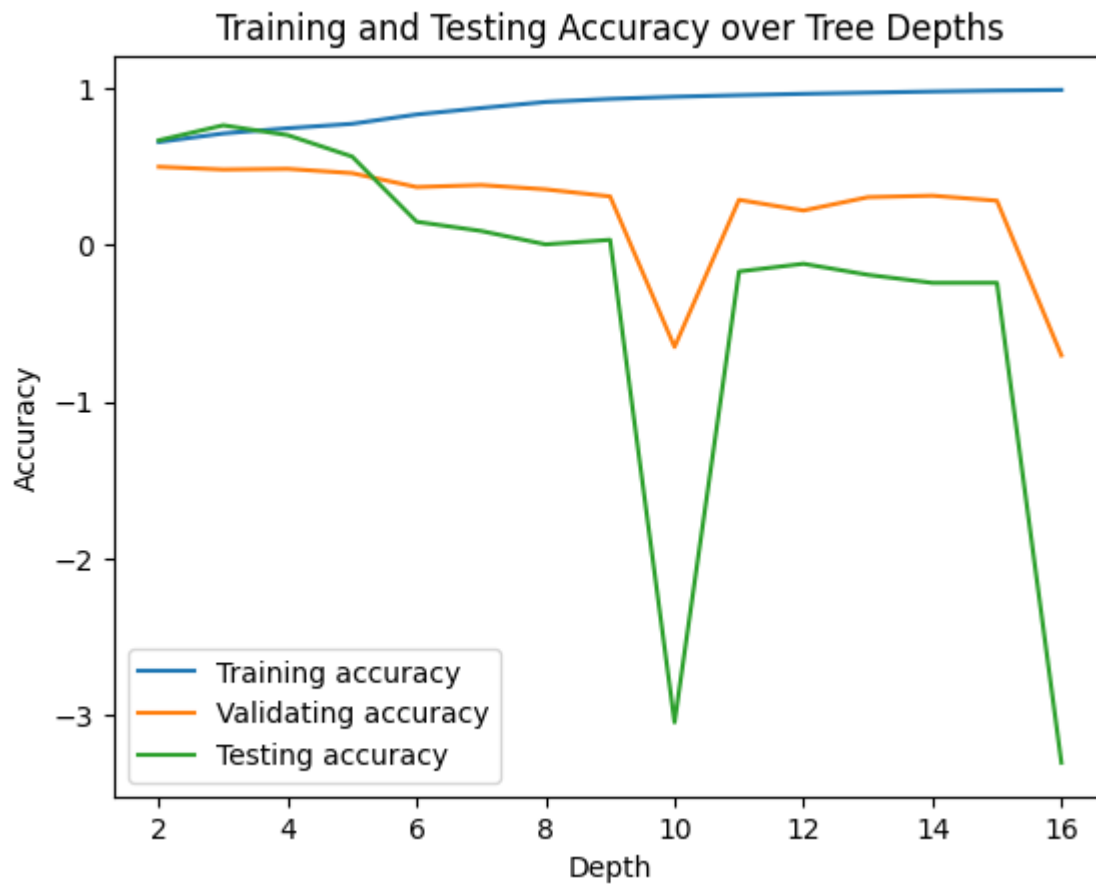
```

import matplotlib.pyplot as plt

plt.plot(tree_depths, train_avg, label='Training accuracy')
plt.plot(tree_depths, valid_avg, label='Validating accuracy')
plt.plot(tree_depths, test_avg, label='Testing accuracy')
plt.xlabel('Depth')
plt.ylabel('Accuracy')
plt.title('Training and Testing Accuracy over Tree Depths')

```

```
plt.legend()  
plt.show()
```



```
# max_index = np.argmax(test_avg)  
max_index = np.argmax(valid_avg)  
dtree = DecisionTreeRegressor(max_depth=tree_depths[3])  
dtree.fit(X_train, y_train)  
dtree.score(X_test, y_test)
```

0.8845159049463991

```
predictions = dtree.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error as MSE
```

```
MSE(y_test, predictions)
```

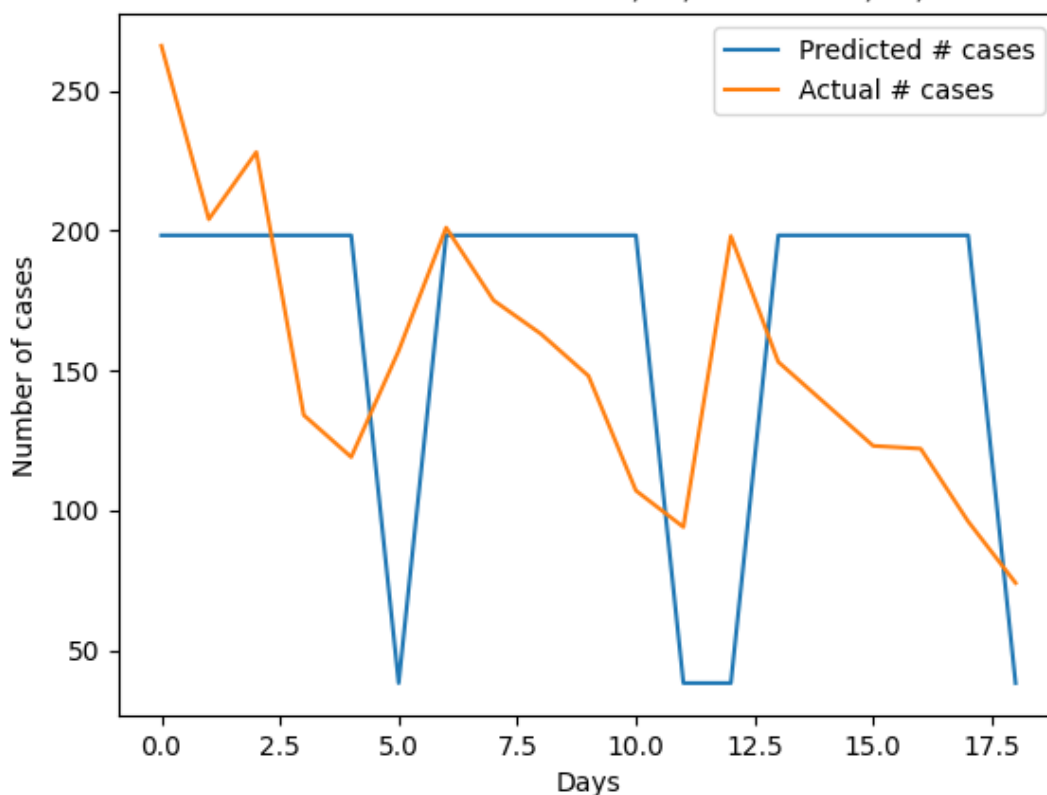
3222.4131115199293

```
pred_county_A = predictions[:,len(time_series['geo_value'].unique())]  
county_A = test.loc[test['geo_value']==6001, columns[-1]]
```

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

```
plt.plot(range(19), pred_county_A, label='Predicted # cases')  
plt.plot(range(19), county_A, label='Actual # cases')  
plt.xlabel('Days')  
plt.ylabel('Number of cases')  
plt.title('Number of Predicted and Actual cases from 02/10/2021 to 02/28/2021 in County 06001')  
plt.legend()  
plt.show()
```

Number of Predicted and Actual cases from 02/10/2021 to 02/28/2021 in County 06001

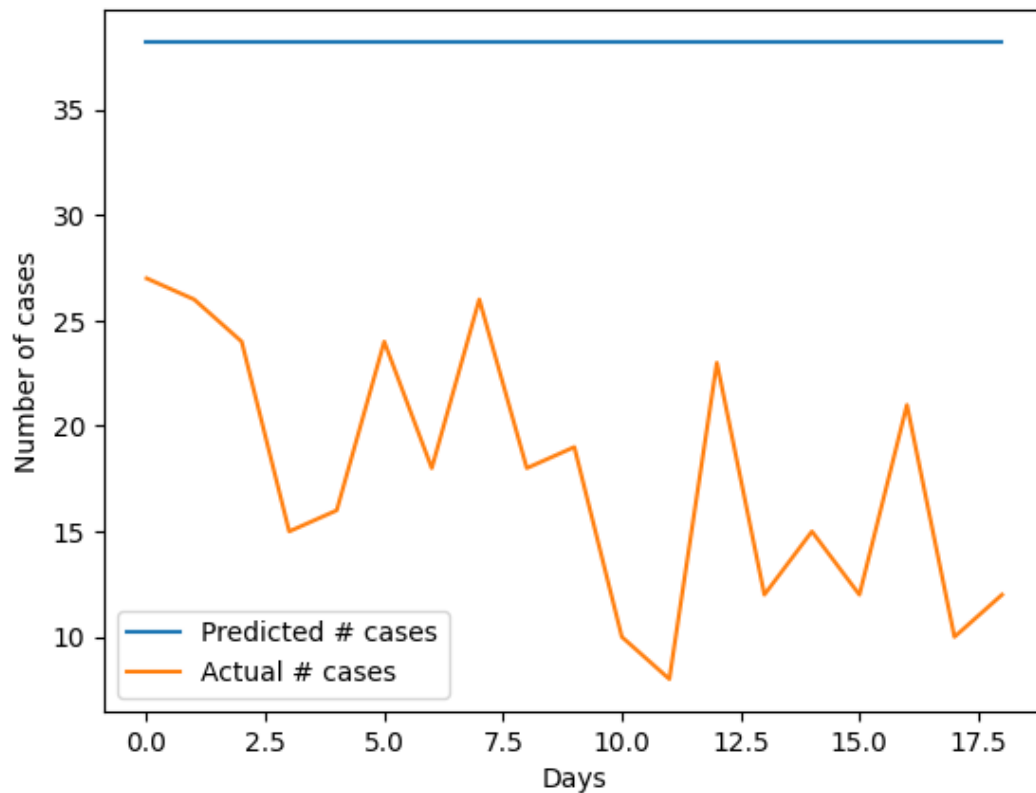


```
pred_county_B = predictions[3::len(time_series['geo_value'].unique())]
county_B = test.loc[test['geo_value']==6007, columns[-1]]
```

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

```
plt.plot(range(19), pred_county_B, label='Predicted # cases')
plt.plot(range(19), county_B, label='Actual # cases')
plt.xlabel('Days')
plt.ylabel('Number of cases')
plt.title('Number of Predicted and Actual cases from 02/10/2021 to 02/28/2021 in County 06007')
plt.legend()
plt.show()
```

Number of Predicted and Actual cases from 02/10/2021 to 02/28/2021 in County 06007



Complex model

```
from sklearn.preprocessing import StandardScaler
```

```
X_train_scaled = StandardScaler().fit_transform(X_train)
X_test_scaled = StandardScaler().fit_transform(X_test)
print('Done')
```

Done

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVR

tscv = TimeSeriesSplit(n_splits=7)
param_grid = {'C': [0.1, 1, 10, 100],
              'gamma': [1, 0.1, 0.01, 0.001],
              'kernel': ['rbf']}
grid = GridSearchCV(estimator=SVR(), cv=tscv, param_grid=param_grid,
                    refit=True, scoring='r2', return_train_score=True)
grid.fit(X_train_scaled, y_train)
print('Done')
```

Done

```
best_model = grid.best_estimator_
print(best_model)
print(grid.best_score_)
```

```
SVR(C=100, gamma=0.1)
0.3727300058934732
```

```
predictions = best_model.predict(X_test_scaled)
```

```
from sklearn.metrics import mean_squared_error as MSE

MSE(y_test, predictions)
```

```
88617.11830536007
```

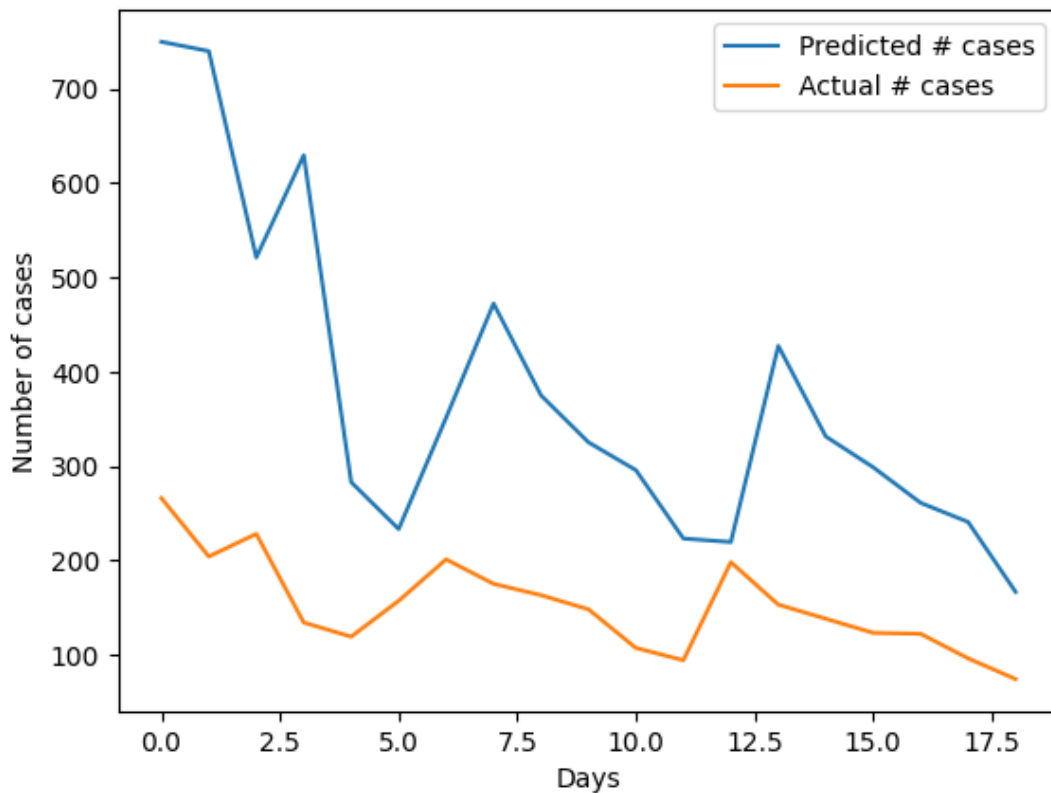
```
pred_county_A = predictions[:,len(time_series['geo_value'].unique())]
county_A = test.loc[test['geo_value']==6001, columns[-1]]
```

```
import matplotlib.pyplot as plt

plt.plot(range(19), pred_county_A, label='Predicted # cases')
plt.plot(range(19), county_A, label='Actual # cases')
plt.xlabel('Days')
plt.ylabel('Number of cases')
plt.title('Number of Predicted and Actual cases from 02/10/2021 to 02/28/2021 in County 06001')
```

```
plt.legend()
plt.show()
```

Number of Predicted and Actual cases from 02/10/2021 to 02/28/2021 in County 06001



```
pred_county_B = predictions[3::len(time_series['geo_value'].unique())]
county_B = test.loc[test['geo_value']==6007, columns[-1]]
```

```
import matplotlib.pyplot as plt

plt.plot(range(19), pred_county_B, label='Predicted # cases')
plt.plot(range(19), county_B, label='Actual # cases')
plt.xlabel('Days')
plt.ylabel('Number of cases')
```

```
plt.title('Number of Predicted and Actual cases from 02/10/2021 to 02/28/2021 in County 06007')  
plt.legend()  
plt.show()
```

Number of Predicted and Actual cases from 02/10/2021 to 02/28/2021 in County 06007

