In [3]:

```
covid_data2 = covid_data2.drop(columns = ['Unnamed: 0', 'Date'])
covid_data2 = covid_data2.iloc[59:21534,:].reset_index().drop(columns = ['ind
covid_data2.columns=["% Confirmed COVID Doctor Visits","% Confirmed COVID Doc
covid_data2
```

Out[3]:

	% Confirmed COVID Doctor Visits	% Confirmed COVID Doctor Visits-1	% Visits with Symptons	% Visits with Symptons -1	% Confirmed COVID doctor Visits	% Confirmed COVID doctor Visits-1	2 % Visits w/Symptoms	w/S
0	0.248550	0.233145	0.866085	0.934454	0.248550	0.233145	0.061340	
1	0.013988	0.011162	0.040125	0.045153	0.013988	0.011162	0.000000	
2	0.612419	0.612419	4.724272	4.724272	0.644757	0.644757	5.107044	
3	0.129362	0.125200	0.228643	0.193900	0.129362	0.125200	0.000000	
4	0.049212	0.053394	0.089703	0.109196	0.049212	0.053394	0.000000	
...	...	...	...	...	...	...	...	
21470	0.860925	0.871385	4.902506	3.866100	0.790400	0.833543	5.107044	
21471	0.798493	0.773726	1.880604	2.285508	0.812549	0.856947	4.471143	
21472	0.525537	0.475692	0.809111	0.840926	0.630288	0.566797	2.605528	
21473	0.302824	0.316138	1.398440	1.634463	0.315021	0.344551	2.444655	
21474	0.378473	0.283863	3.703351	4.677999	0.370449	0.315869	3.067841	

21475 rows × 13 columns

In [4]:

```
covid_data2.isnull().sum()
```

Out[4]:

```
% Confirmed COVID Doctor Visits        0
% Confirmed COVID Doctor Visits-1      0
% Visits with Symptons                 0
% Visits with Symptons -1              0
% Confirmed COVID doctor Visits        0
% Confirmed COVID doctor Visits-1      0
2 % Visits w/Symptoms                  0
2 % Visits w/Symptoms-1                0
% New Visits COVID- associated         0
% New Visits COVID- associated-1       0
Ground Truth Cases                     0
Ground Truth Cases-1                   0
Ground Truth Cases+1                   0
dtype: int64
```

In [5]:

```
X = covid_data2.iloc[:, 0:13]
y = covid_data2.iloc[:, -1]
```

In [6]:

```
from sklearn.model_selection import TimeSeriesSplit

#Manually split data chronologically; first 80% of total data is train and th
test_size = int(0.2*(len(covid_data2)))
train_size = int(0.8*(len(covid_data2)))

#Use this for Cross-Validation process
tscv = TimeSeriesSplit(n_splits = 5, max_train_size=train_size, test_size = 0
```

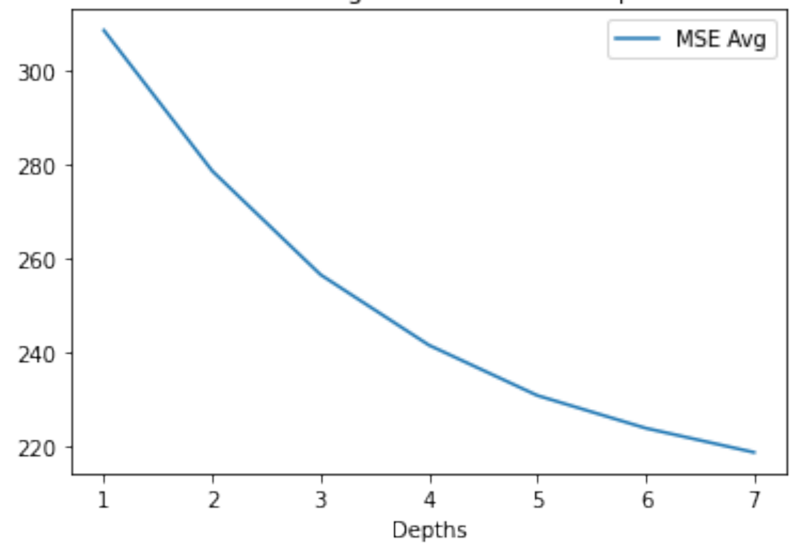
In [7]:

```
X_train=X.iloc[:train_size,:]
y_train=y.iloc[:train_size,]
X_test=X.iloc[train_size:,:]
y_test=y.iloc[train_size:,]
```

In [10]:

```
#DECISION TREE REGRESSOR


from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics
from sklearn.metrics import mean_squared_error
import math
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_validate
import matplotlib.pyplot as plt
import numpy as np

tscv = TimeSeriesSplit()

dt_mse_arr = []
dt_r2_arr = []
dt_y_pred_folds = []
dt_mse_avg, dt_r2_avg= [],[]
#use depths 2-10
depths = [1,2,3,4,5,6,7]
#bestAcc = 0

#validation set
for depth_value in depths:
    for fold, (train_index, val_index) in enumerate(tscv.split(X_train)):
        dt_X_tr, dt_X_val = X_train.iloc[train_index], X_train.iloc[val_index
        dt_y_tr, dt_y_val = y_train.iloc[train_index], y_train.iloc[val_index
        tree=DecisionTreeRegressor(max_depth = depth_value)
        tree.fit(dt_X_tr, dt_y_tr)
        for loop in range(500):
            dt_y_pred = tree.predict(dt_X_val)
            dt_curr_error_val = mean_squared_error(dt_y_val,dt_y_pred,squared
            #print("MSE Score for loop", loop, "in fold", fold, "w/ depth", d
            dt_mse_arr.append(dt_curr_error_val)
            dt_r2 = r2_score(dt_y_val, dt_y_pred)
            #print("R^2 Score for loop", loop, "in fold", fold, "w/ depth", d
            dt_r2_arr.append(dt_r2)
    dt_mse_avg.append(np.mean(dt_mse_arr))
    print("AVG MSE for depth", depth_value, ":", np.mean(dt_mse_arr))
    dt_r2_avg.append(np.mean(dt_r2_arr))
    print("AVG R^2 for depth", depth_value, ":", np.mean(dt_r2_arr))
    # store the best tree
    # if np.mean(dt_r2_arr) > bestAcc:
    #     bestAcc = np.mean(dt_r2_arr)
    #     bestTree = tree

print("Validation Score:")
print("Average RMSE:", np.mean(dt_mse_avg))
print("Average R^2:", np.mean(dt_r2_avg))

plt.plot(dt_mse_avg, label = "MSE Avg")
plt.legend(loc = 'best')
plt.xticks(range(7), [1,2,3,4,5,6,7])
plt.xlabel("Depths")
plt.title("Plot of Average MSE Across All Depths")
plt.show()


plt.plot(dt_r2_avg, label = "R^2 Avg")
plt.legend(loc = 'best')
plt.xticks(range(7), [1,2,3,4,5,6,7])
plt.xlabel("Depths")
plt.title("Figure C: Plot of Average R^2 Across All Depths")
plt.show()
```

```
AVG MSE for depth 1 : 308.5086750243539
AVG R^2 for depth 1 : 0.5686657779528408
AVG MSE for depth 2 : 278.60760856898554
AVG R^2 for depth 2 : 0.6549339228518158
AVG MSE for depth 3 : 256.55205184724406
AVG R^2 for depth 3 : 0.7156203820526575
AVG MSE for depth 4 : 241.58160969387967
AVG R^2 for depth 4 : 0.7531583233589315
AVG MSE for depth 5 : 230.88168919946602
AVG R^2 for depth 5 : 0.7780679372710482
AVG MSE for depth 6 : 223.93666093683703
AVG R^2 for depth 6 : 0.7936473205375816
AVG MSE for depth 7 : 218.79449969133435
AVG R^2 for depth 7 : 0.8048723660251778
Validation Score:
Average RMSE: 251.26611356601435
Average R^2: 0.7241380042928647
```



Plot of Average MSE Across All Depths

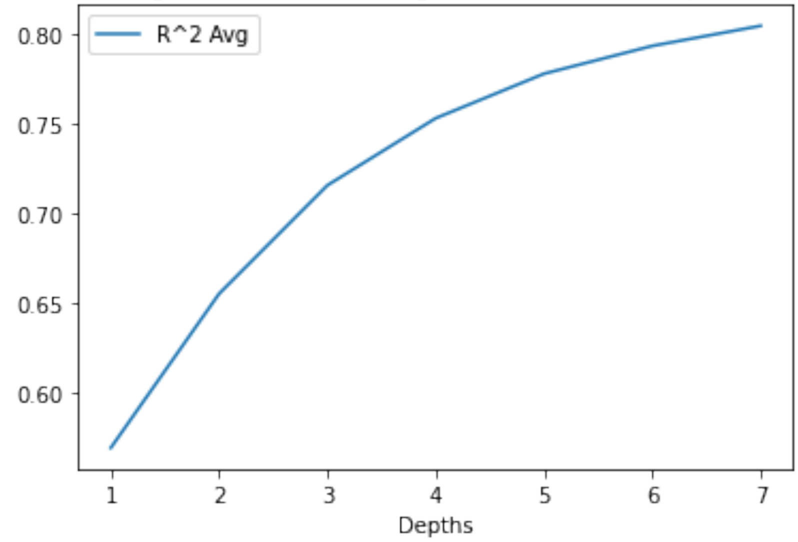


Figure C: Plot of Average R^2 Across All Depths

In [11]:

```
tree = DecisionTreeRegressor(max_depth = 4).fit(X_train, y_train)
y_pred_max_depth= tree.predict(X_test)
tree_mse_max_depth = mean_squared_error(y_test, y_pred_max_depth, squared=Fal
tree_r2_max_depth = r2_score(y_test,y_pred_max_depth)

print("Testing Scores:")
print("Test MSE:", tree_mse_max_depth)
print("Test R^2:", tree_r2_max_depth)
```

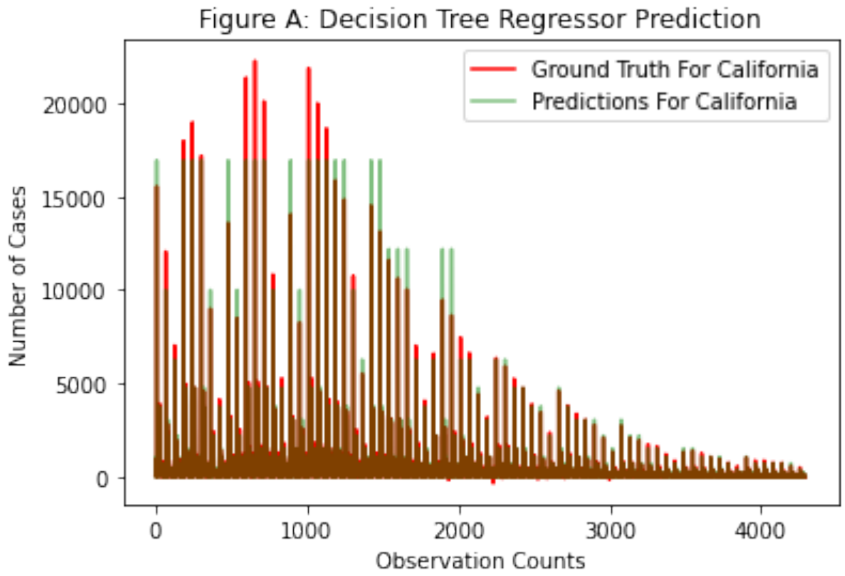
```
Testing Scores:
Test MSE: 220.38224960674708
Test R^2: 0.9736006364323077
```

In [12]:

```
ground_y=covid_data2.iloc[train_size:,-1:]
ground_y_graph = ground_y.reset_index()
ground_y_graph = ground_y_graph.drop(columns = ['index'])
```

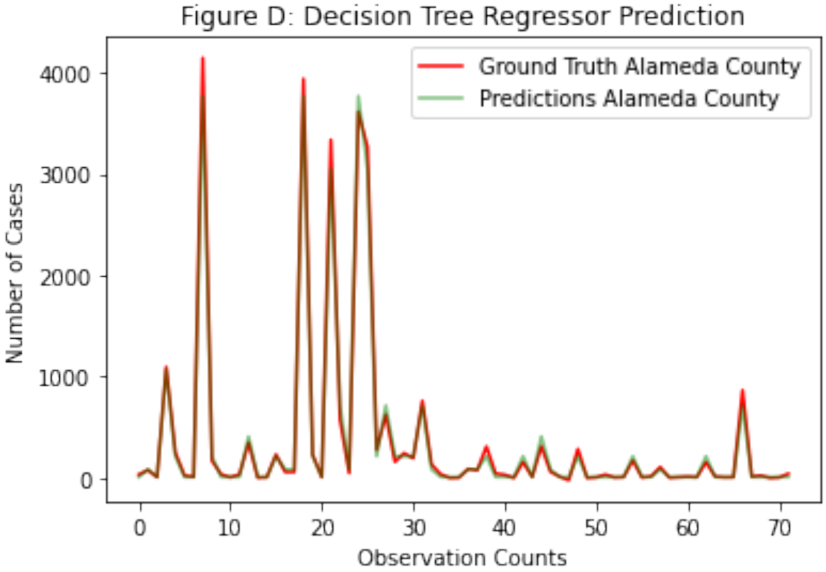
In [118…

```
plt.plot(ground_y_graph, color='red', label= "Ground Truth For California")
plt.plot(y_pred_max_depth, color='green', label="Predictions For California",
plt.legend(loc = 'best')
plt.title("Figure A: Decision Tree Regressor Prediction")
plt.xlabel('Observation Counts')
plt.ylabel('Number of Cases')
plt.show()
```



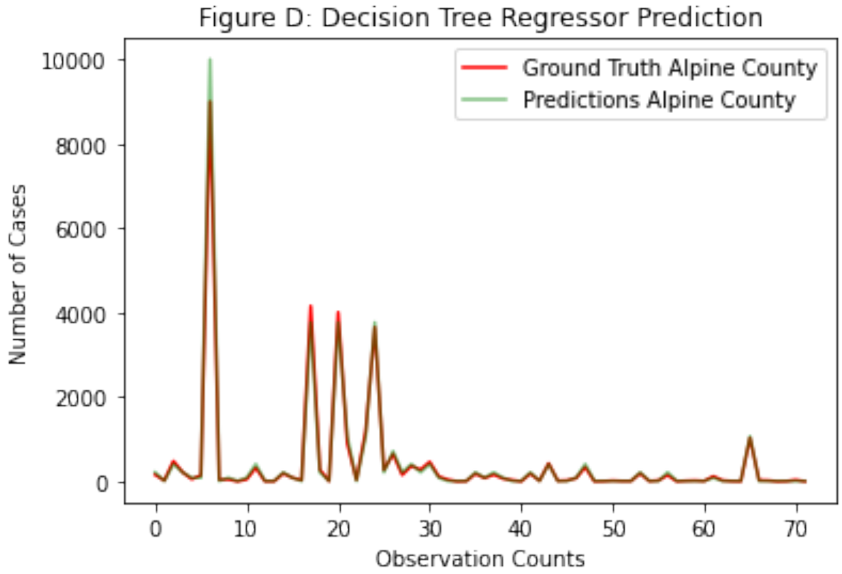
In [111…

```
ground_test=ground_y_graph
ground_test2=ground_test.values.tolist()
ground_test3=ground_test2[1::60]

pred_test=y_pred_max_depth
pred_test2=pred_test.tolist()
pred_test3=pred_test2[1::60]


plt.plot(ground_test3, color='red', label= "Ground Truth Alameda County")
plt.plot(pred_test3, color='green', label="Predictions Alameda County", alpha
plt.legend(loc = 'best')
plt.title("Figure D: Decision Tree Regressor Prediction")
plt.xlabel('Observation Counts')
plt.ylabel('Number of Cases')
plt.show()
```



In [116…

```
#alpine county
ground_test=ground_y_graph
ground_test2=ground_test.values.tolist()
ground_test3=ground_test2[2::60]

pred_test=y_pred_max_depth
pred_test2=pred_test.tolist()
pred_test3=pred_test2[2::60]


plt.plot(ground_test3, color='red', label= "Ground Truth Alpine County")
plt.plot(pred_test3, color='green', label="Predictions Alpine County", alpha
plt.legend(loc = 'best')
plt.title("Figure D: Decision Tree Regressor Prediction")
plt.xlabel('Observation Counts')
plt.ylabel('Number of Cases')
plt.show()
```



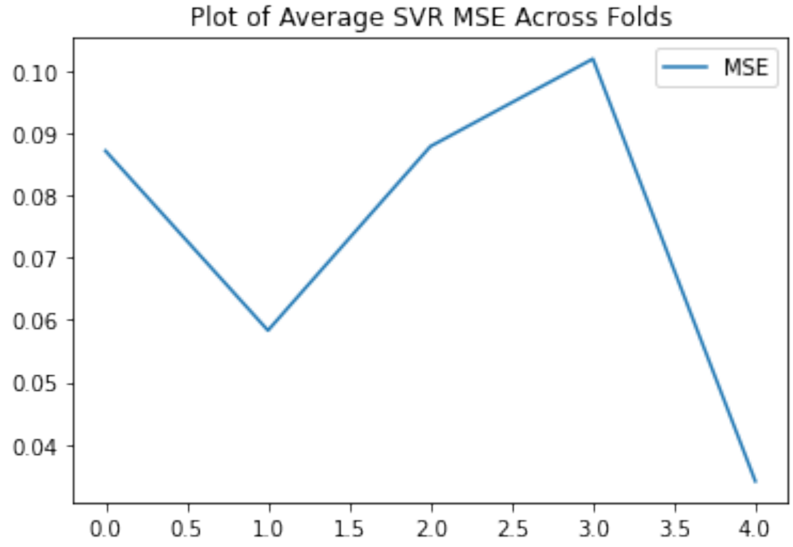
In [49]:

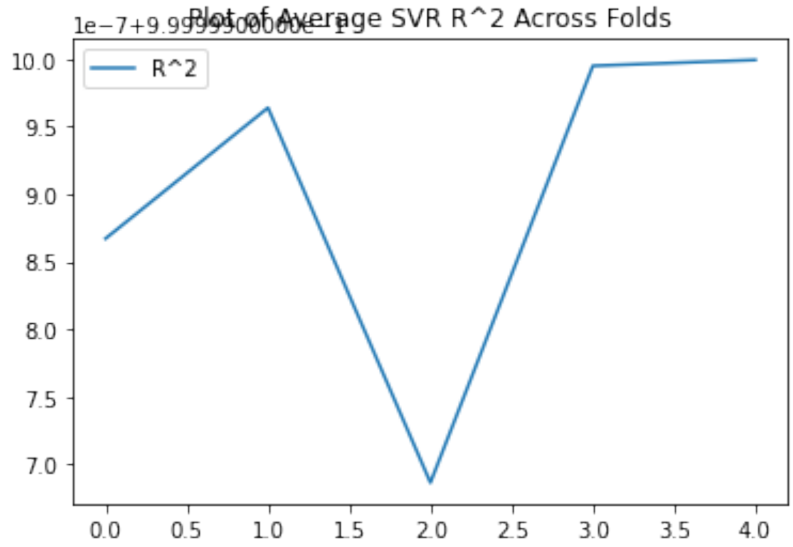
```
ground_y=covid_data2.iloc[train_size:,-1:]
ground_y_graph = ground_y.reset_index()
ground_y_graph = ground_y_graph.drop(columns = ['index'])
```

In [50]:

```
#SUPPORT VECTOR MACHINE

from sklearn.metrics import accuracy_score
from sklearn.svm import SVR



tscv = TimeSeriesSplit()

mse_arr=[]
r2_arr=[]

for fold, (train_index, test_index) in enumerate(tscv.split(X)):
    # print("Fold: {}".format(fold))
    # print("TRAIN indices:", train_index, "\n", "TEST indices:", test_index)
    # print("\n")
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    svregressor = SVR(kernel = 'linear')
    svregressor.fit(X_train, y_train)
    y_prediction = svregressor.predict(X_test)
    curr_error_test = mean_squared_error(y_test,y_prediction,squared=False)
    mse_arr.append(curr_error_test)
    r2 = r2_score(y_test, y_prediction)
    r2_arr.append(r2)
    mse_avg= np.mean(mse_arr)
    r2_avg= np.mean(r2_arr)

print("SVR R^2 Average:", r2_avg)
print("SVR MSE Average:", mse_avg)

plt.plot(mse_arr, label = "MSE")
plt.title("Plot of Average SVR MSE Across Folds")
plt.legend(loc = 'best')
plt.show()

plt.plot(r2_arr, label = "R^2")
plt.title("Plot of Average SVR R^2 Across Folds")
plt.legend(loc = 'best')
plt.show()
```

```
SVR R^2 Average: 0.999999902269329
SVR MSE Average: 0.07385861141497
```





In [51]:

```python
from sklearn.preprocessing import StandardScaler


sc_X = StandardScaler()
sc_y = StandardScaler()
X_train_sc = sc_X.fit_transform(X_train)
y_train_sc = sc_y.fit_transform(np.array(y_train).reshape(-1,1))
X_train_sc = pd.DataFrame(X_train_sc)
y_train_sc = pd.DataFrame(y_train_sc)
y_train_sc = y_train_sc.squeeze()
```

In [94]:

```python
X_train=X.iloc[:train_size,:]
y_train=y.iloc[:train_size,]
X_test=X.iloc[train_size:,:]
y_test=y.iloc[train_size:,]
```

```python
from sklearn.tree import DecisionTreeClassifi
```

In [102…

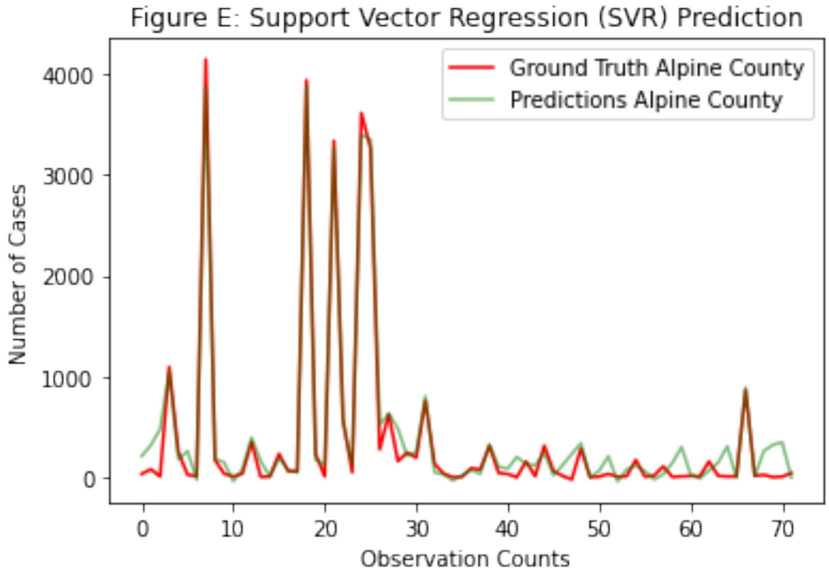
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics
from sklearn.metrics import mean_squared_error
import math
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_validate
import numpy as np
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler

tscv = TimeSeriesSplit()

score_arr = []
r2_arr = []
y_pred_folds = []
r2_avg=[]

sc_X = StandardScaler()
sc_y = StandardScaler()
X_train_sc = sc_X.fit_transform(X_train)
y_train_sc = sc_y.fit_transform(np.array(y_train).reshape(-1,1))
X_train_sc = pd.DataFrame(X_train_sc)
y_train_sc = pd.DataFrame(y_train_sc)
y_train_sc = y_train_sc.squeeze()
X_test_sc = sc_X.fit_transform(X_test)
y_test_sc = sc_y.fit_transform(np.array(y_test).reshape(-1,1))
y_test_sc = y_test_sc.squeeze()

#validation set
for fold, (train_index, val_index) in enumerate(tscv.split(X_train_sc)):
    X_tr, X_val = X_train_sc.iloc[train_index], X_train_sc.iloc[val_index]
    y_tr, y_val = y_train_sc.iloc[train_index], y_train_sc.iloc[val_index]
    svregressor = SVR(kernel = 'rbf')
    svregressor.fit(X_tr, y_tr)
    y_pred = svregressor.predict(X_val)
    y_pred_folds.append(y_pred)
    currAcc_train = mean_squared_error(y_val,y_pred,squared=False)
    score_arr.append(currAcc_train)
    print("MSE for fold", fold, ":", currAcc_train)
    r2 = r2_score(y_val, y_pred)
    r2_avg.append(r2)
    print("R^2 for fold", fold, ":", r2)

#test results

svregressor = SVR(kernel = 'rbf').fit(X_train_sc, y_train_sc)
y_pred3 = svregressor.predict(X_test_sc)
y_pred3 = sc_y.inverse_transform(np.array(y_pred3).reshape(-1,1))

mse = mean_squared_error(y_test, y_pred3, squared=False)
r2d2 = r2_score(y_test,y_pred3)
```

```
print("Validation Score:")
print("Average RMSE:", np.mean(score_arr))
print("Average R^2", np.mean(r2_avg))

print("Testing Scores:")
print("Test MSE:", mse)
print("Test R^2:", r2d2)
```

```
MSE for fold 0 : 0.2007194869190263
R^2 for fold 0 : 0.46436069285035975
MSE for fold 1 : 0.5859830781773039
R^2 for fold 1 : 0.2720013429737115
MSE for fold 2 : 0.08575828029829051
R^2 for fold 2 : 0.9475381088677851
MSE for fold 3 : 0.11222922868149768
R^2 for fold 3 : 0.8655498001133816
MSE for fold 4 : 2.066668781579402
R^2 for fold 4 : 0.13306183714979658
Validation Score:
Average RMSE: 0.6102717711311041
Average R^2 0.5365023563910069
Testing Scores:
Test MSE: 913.1446236425514
Test R^2: 0.5467692513610569
```

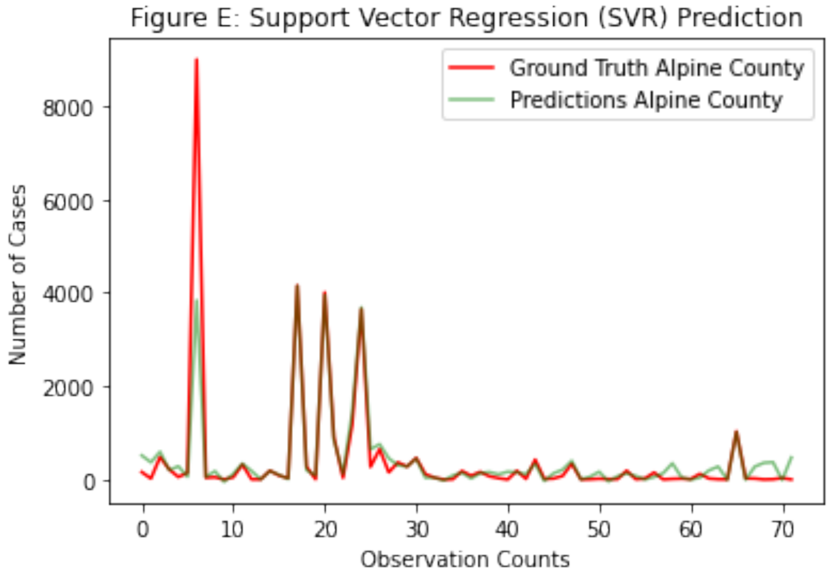
In [109…

```
ground_test=ground_y_graph
ground_test2=ground_test.values.tolist()
ground_test3=ground_test2[1::60]

pred_test8=y_pred3
pred_test2=pred_test8.tolist()
pred_test5=pred_test2[1::60]


plt.plot(ground_test3, color='red', label= "Ground Truth Alpine County")
plt.plot(pred_test5, color='green', label="Predictions Alpine County", alpha
plt.legend(loc = 'best')
plt.title("Figure E: Support Vector Regression (SVR) Prediction")
plt.xlabel('Observation Counts')
plt.ylabel('Number of Cases')
plt.show()
```



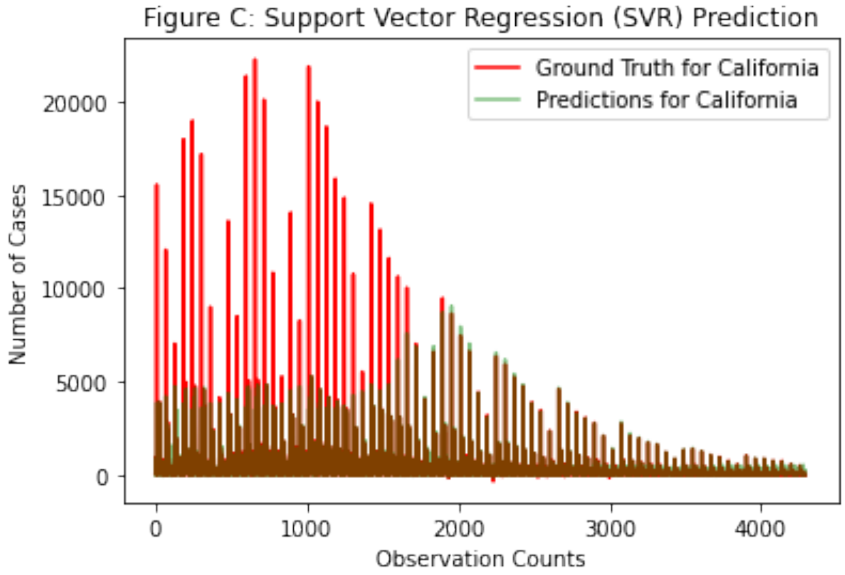
In [117…

```
ground_test=ground_y_graph
ground_test2=ground_test.values.tolist()
ground_test3=ground_test2[2::60]

pred_test8=y_pred3
pred_test2=pred_test8.tolist()
pred_test5=pred_test2[2::60]


plt.plot(ground_test3, color='red', label= "Ground Truth Alpine County")
plt.plot(pred_test5, color='green', label="Predictions Alpine County", alpha
plt.legend(loc = 'best')
plt.title("Figure E: Support Vector Regression (SVR) Prediction")
plt.xlabel('Observation Counts')
plt.ylabel('Number of Cases')
plt.show()
```



In [119…

```
plt.plot(ground_y_graph, color='red', label= "Ground Truth for California")
plt.plot(y_pred3, color='green', label="Predictions for California", alpha =
plt.legend(loc = 'best')
plt.title("Figure C: Support Vector Regression (SVR) Prediction")
plt.xlabel('Observation Counts')
plt.ylabel('Number of Cases')
plt.show()
```



In [120…

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics
from sklearn.metrics import mean_squared_error
import math
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_validate
import numpy as np
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler

tscv = TimeSeriesSplit()

score_arr = []
r2_arr = []
y_pred_folds = []
r2_avg=[]

sc_X = StandardScaler()
sc_y = StandardScaler()
X_train_sc = sc_X.fit_transform(X_train)
y_train_sc = sc_y.fit_transform(np.array(y_train).reshape(-1,1))
X_train_sc = pd.DataFrame(X_train_sc)
y_train_sc = pd.DataFrame(y_train_sc)
y_train_sc = y_train_sc.squeeze()
X_test_sc = sc_X.fit_transform(X_test)
```

```
y_test_sc = sc_y.fit_transform(np.array(y_test).reshape(-1,1))
y_test_sc = y_test_sc.squeeze()

#validation set
for fold, (train_index, val_index) in enumerate(tscv.split(X_train_sc)):
    X_tr, X_val = X_train_sc.iloc[train_index], X_train_sc.iloc[val_index]
    y_tr, y_val = y_train_sc.iloc[train_index], y_train_sc.iloc[val_index]
    svregressor = SVR(kernel = 'linear')
    svregressor.fit(X_tr, y_tr)
    y_pred = svregressor.predict(X_val)
    y_pred_folds.append(y_pred)
    currAcc_train = mean_squared_error(y_val,y_pred,squared=False)
    score_arr.append(currAcc_train)
    print("MSE for fold", fold, ":", currAcc_train)
    r2 = r2_score(y_val, y_pred)
    r2_avg.append(r2)
    print("R^2 for fold", fold, ":", r2)

#test results

svregressor = SVR(kernel = 'linear').fit(X_train_sc, y_train_sc)
y_pred4 = svregressor.predict(X_test_sc)
y_pred4 = sc_y.inverse_transform(np.array(y_pred4).reshape(-1,1))

mse = mean_squared_error(y_test, y_pred3, squared=False)
r2d2 = r2_score(y_test,y_pred3)

print("Validation Score:")
print("Average RMSE:", np.mean(score_arr))
print("Average R^2", np.mean(r2_avg))

print("Testing Scores:")
print("Test MSE:", mse)
print("Test R^2:", r2d2)
```

```
MSE for fold 0 : 0.08104706168599003
R^2 for fold 0 : 0.9126691047896515
MSE for fold 1 : 0.06511852679666473
R^2 for fold 1 : 0.9910097956785107
MSE for fold 2 : 0.03675707711109297
R^2 for fold 2 : 0.9903622922343375
MSE for fold 3 : 0.03555033155727283
R^2 for fold 3 : 0.98650922786529
MSE for fold 4 : 0.06742312071448947
R^2 for fold 4 : 0.9990772924736041
Validation Score:
Average RMSE: 0.05717922357310201
Average R^2 0.9759255426082787
Testing Scores:
Test MSE: 913.1446236425514
Test R^2: 0.5467692513610569
```

In [122…

```
plt.plot(ground_y_graph, color='red', label= "Ground Truth for California")
plt.plot(y_pred3, color='green', label="Predictions for California", alpha =
plt.legend(loc = 'best')
plt.title("Figure B: SVR Prediction")
plt.xlabel('Observation Counts')
plt.ylabel('Number of Cases')
plt.show()
```

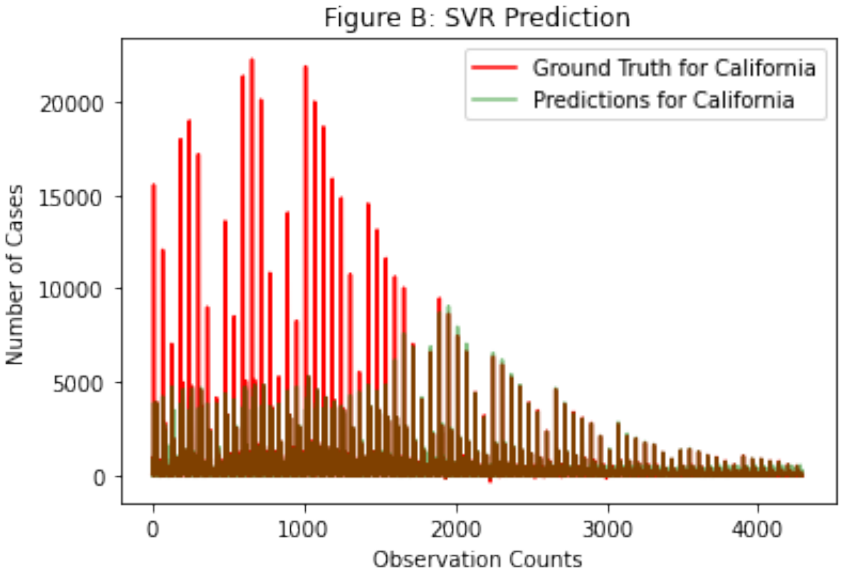


Figure B: SVR Prediction
Ground Truth for California
Predictions for California
Number of Cases
Observation Counts