Assignment_5

March 22, 2022

[ ]:
```
# Rohit Ranjan
# 20CS30066
# Assignment 5
```

[1]:
```
#import commands
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (80,80)
import pandas as pd
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV

#xgboost
import xgboost as xgb

#SVM
from sklearn.svm import SVR
```

[2]:
```
#datafile provided
filename = "S_n_I_A_N_P_An_Io_noaa2.txt"
```

[3]:
```
#preprocessing the datafile
rows=[]
for line in open(filename):
    curr_row = list(map(float, line.split()))
    rows.append(curr_row)
```

[4]:
```
len(rows)
```

[4]: 870

[5]:
```
df = pd.DataFrame(rows)
df.columns = ['SAM','nino','ISMR','AMO','NAO','PDO','At-nino','IOD']
```

[6]:
```
cols_ordered = ['SAM','nino','AMO','NAO','PDO','At-nino','IOD','ISMR']
df = df[cols_ordered]
```

[7]:
```
df.head(5)
```

[7]:
```
        SAM      nino       AMO       NAO       PDO   At-nino       IOD  \
0  0.599257  0.167651  0.484010  0.000837  1.084100  0.560382 -0.295856
1  0.663879  0.212788  0.603261 -0.317969 -1.465790  1.189540 -0.032952
2 -0.428772  0.336627  0.609364  0.423296  0.114019  0.783738 -0.644708
3  0.513283  0.575112  0.219494  0.352545  1.807430  0.560395 -0.295477
4 -0.260801  0.647786  0.040233 -0.528239 -0.108867  0.727379  0.044719

       ISMR
0  10.24380
1  44.16110
2   4.76807
3 -63.58770
4  12.57720
```

[8]:
```
#Experiments with xgBoost
```

[9]:
```
true_values = []
pred_values = []
rmse_values = []

for lag in range(1,11):
    # forming features using lag
    y = df['ISMR'][lag:]
    X = df.iloc[:-1*lag,:]

    # converting to efficient data structure
    data_dmatrix = xgb.DMatrix(data=X,label=y)

    # intialising xgboost regressor object and fitting on data
    xg_reg = xgb.XGBRegressor(colsample_bytree = 0.3, learning_rate = 0.
↪1,max_depth = 5, alpha = 10, n_estimators = 1000)
    xg_reg.fit(X,y)

    #generating predictions
    preds = xg_reg.predict(X)

    # calculating error to help as compare lag performance
    rmse = np.sqrt(mean_squared_error(y, preds))

    # appending values to a list for later use
    true_values.append(y)
    pred_values.append(preds)
    rmse_values.append((rmse))
```

[10]:
```
rmse_table = pd.DataFrame(list(np.array(rmse_values).
↪reshape((len(rmse_values),-1)).T))
rmse_table.columns = [("Lag_"+str(i)) for i in range(1,11)]
```

[11]:
```
rmse_table
```

[11]:
```
      Lag_1     Lag_2     Lag_3    Lag_4     Lag_5     Lag_6     Lag_7  \
0  1.586117  1.700843  1.797371  1.75844  1.742344  1.723301  1.743826

      Lag_8     Lag_9    Lag_10
0  1.765855  1.814621  1.634117
```

[12]:
```
# computing table for different lags
table = pd.DataFrame()
table['True ISMR'] = df['ISMR']
```

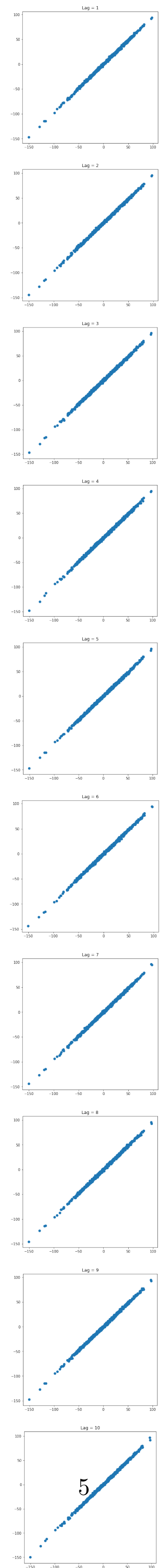
[13]:
```
# padding the lag columns
for i in range(len(pred_values)):
    pred_lag = list(pred_values[i])
    for i in range(df.shape[0]-len(pred_lag)):
        pred_lag.insert(0,np.NaN)
    table['Lag_'+str(i+1)] = pred_lag
```

[14]:
```
table[0:20]
```

[14]:
```
    True ISMR      Lag_1      Lag_2      Lag_3      Lag_4      Lag_5  \
0   10.243800        NaN        NaN        NaN        NaN        NaN
1   44.161100  43.150269        NaN        NaN        NaN        NaN
2    4.768070   5.338588   4.439160        NaN        NaN        NaN
3  -63.587700 -62.148609 -62.418106 -58.706940        NaN        NaN
4   12.577200  12.076989  11.821412  11.542588  13.834686        NaN
5  -41.577200 -41.952026 -38.927246 -38.944092 -42.661339 -38.924084
6   -9.546880  -9.879082 -10.141637  -9.084824  -9.147939  -9.122007
7   25.370500  24.726170  23.547773  23.525265  24.865967  24.532143
8   18.277400  18.698280  17.388618  16.878601  20.062557  17.476351
9    2.221670   1.521839   2.148856   3.221498   1.718592   0.318806
10  16.886600  17.196455  16.851954  17.330164  16.662622  16.647011
11   0.132196   0.001765   1.290223   2.877347   0.691788   0.578885
12 -10.537500  -9.827923  -9.441161 -12.002758  -9.992893 -10.177773
13 -50.820200 -48.424641 -46.539761 -48.758595 -48.716595 -50.296795
14  -8.513200  -6.482193  -8.650455  -8.616445  -8.550304  -7.871543
15 -28.769000 -27.899918 -25.707705 -28.510756 -27.202774 -27.854412
16  -5.404050  -5.391354  -4.667356  -5.478644  -4.768011  -5.960122
17 -17.658400 -17.083843 -17.908487 -16.732082 -15.841544 -17.970730
18  14.971800  16.190086  14.987142  14.808299  14.542233  14.397070
19  64.089200  62.903187  64.037880  62.913769  63.030930  62.196003
```

	Lag_6	Lag_7	Lag_8	Lag_9	Lag_10
0	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN
6	-7.095208	NaN	NaN	NaN	NaN
7	24.191381	25.250797	NaN	NaN	NaN
8	16.491549	18.776773	17.648241	NaN	NaN
9	1.528083	3.899342	1.213346	1.961416	NaN
10	17.723703	16.984150	16.306791	16.890469	17.297546
11	1.266631	1.552985	2.110994	1.315464	0.438608
12	-11.356400	-11.348008	-10.918290	-11.448859	-10.131416
13	-47.985355	-46.207077	-48.974602	-47.702118	-49.232628
14	-7.291180	-7.087028	-7.215755	-8.229162	-8.708284
15	-26.521212	-25.365711	-28.066278	-26.121048	-26.770824
16	-6.601791	-5.548816	-4.540854	-5.208636	-4.742649
17	-17.766972	-15.611817	-16.529551	-17.022909	-16.926622
18	14.450985	14.158209	13.296273	14.539018	13.520664
19	63.259460	64.154404	62.819069	61.924686	59.933323

[15]:
```
# plotting scatter plots
fig, axs = plt.subplots(10)
for i in range(10):
    axs[i].scatter(true_values[i],pred_values[i])
    axs[i].xlabel = 'True Values'
    axs[i].ylabel = 'Predicted Values'
    axs[i].set_title("Lag = "+str(i+1))
    axs[i].set_aspect('equal', adjustable='box')
```


Lag = 1
Lag = 2
Lag = 3
Lag = 4
Lag = 5
Lag = 6
Lag = 7
Lag = 8
Lag = 9
Lag = 10

[16]:
```
# The best lag in drivers is 1 from our analysis. From the plotted graphs that␣
↪xgboost trained with lag=1 dataset gives preds that most closely resemble␣
↪the true values. This leads to the graoh being closest to a staright line␣
↪among all 10 lags.
```

[17]:
```
# SVM - Linear Kernel
```

[18]:
```
true_values = []
pred_values = []
rmse_values = []
for lag in range(1,11):
    # forming features using lag
    y = df['ISMR'][lag:]
    X = df.iloc[:-1*lag,:]

    # intialising SVM regressor object and fitting on data
    svr = SVR(kernel ='linear', C=1.0, epsilon=0.1)
    svr.fit(X,y)

    #generating predictions
    preds = svr.predict(X)

    # calculating error to help as compare lag performance
    rmse = np.sqrt(mean_squared_error(y, preds))

    # appending values to a list for later use
    true_values.append(y)
    pred_values.append(preds)
    rmse_values.append((rmse))
```

[19]:
```
rmse_table = pd.DataFrame(list(np.array(rmse_values).
↪reshape((len(rmse_values),-1)).T))
rmse_table.columns = [("Lag_"+str(i)) for i in range(1,11)]
```

[20]:
```
rmse_table
```

[20]:

	Lag_1	Lag_2	Lag_3	Lag_4	Lag_5	Lag_6	Lag_7	\
0	31.550819	32.17455	32.850646	33.040213	32.987998	32.984692	32.878436	

	Lag_8	Lag_9	Lag_10
0	32.752821	32.840307	32.994115

[21]:
```
table = pd.DataFrame()
table['True ISMR'] = df['ISMR']
```

[22]:
```python
for i in range(len(pred_values)):
    pred_lag = list(pred_values[i])
    for i in range(df.shape[0]-len(pred_lag)):
        pred_lag.insert(0,np.NaN)
    table['Lag_'+str(i+1)] = pred_lag
```

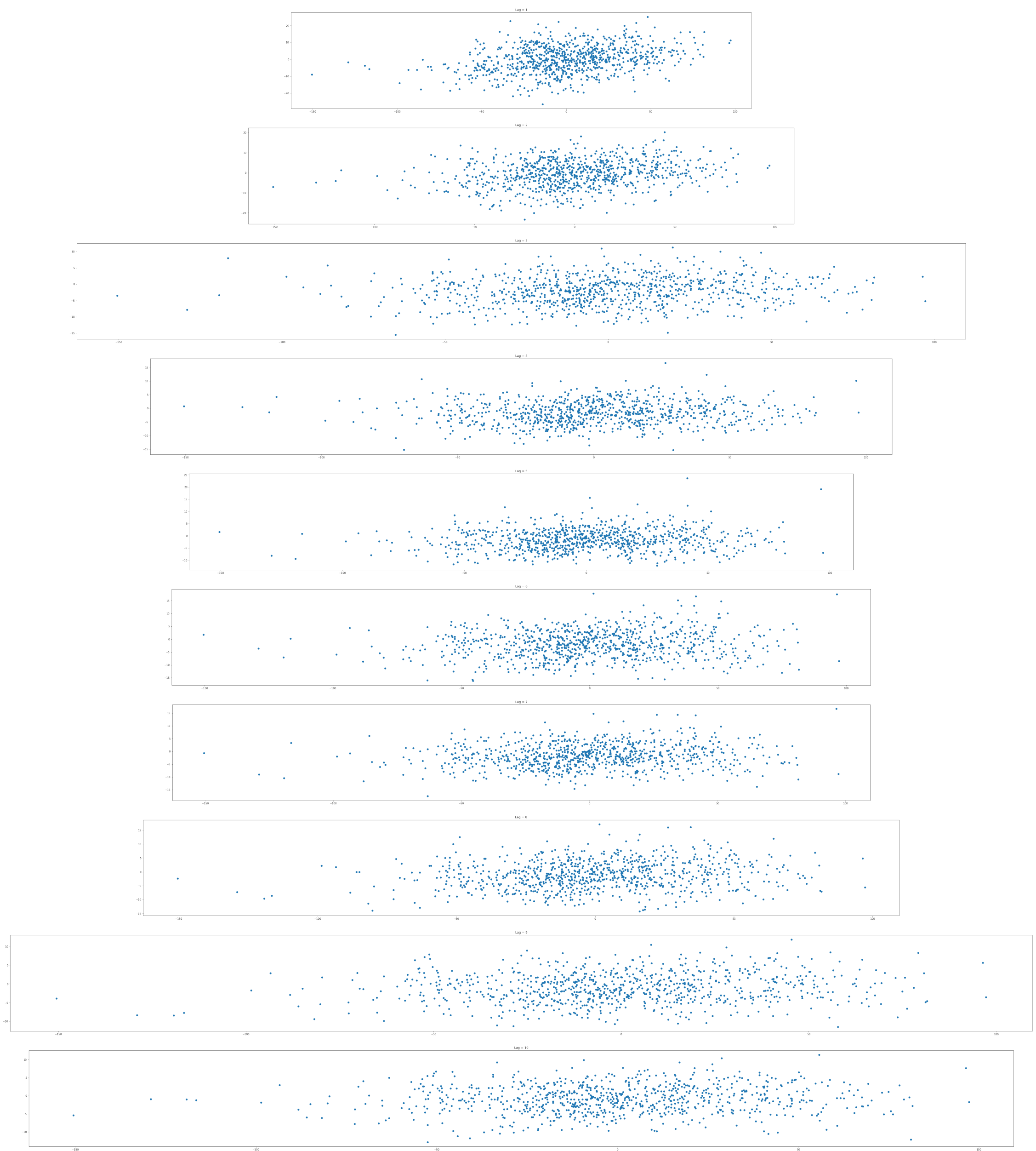
[23]:
```python
table[0:20]
```

[23]:
```
     True ISMR      Lag_1     Lag_2     Lag_3     Lag_4     Lag_5     Lag_6  \
0    10.243800        NaN       NaN       NaN       NaN       NaN       NaN
1    44.161100   0.823180       NaN       NaN       NaN       NaN       NaN
2     4.768070   4.867730  1.860854       NaN       NaN       NaN       NaN
3   -63.587700  -2.126838  6.851839  1.727825       NaN       NaN       NaN
4    12.577200 -11.990512  0.269369  3.652074  5.455071       NaN       NaN
5   -41.577200  -5.372353 -8.518099  1.674918  5.052237  2.126606       NaN
6    -9.546880  -5.253140 -3.546155 -1.582393  4.772103  6.275801  3.273256
7    25.370500   6.234563 -5.974274 -1.381092 -0.262685  4.042350  2.506127
8    18.277400   8.497448  2.104331 -3.466746  1.935317  0.441458  7.155424
9     2.221670  -3.681849  7.646596 -0.605787 -8.325702  1.718356  5.751010
10   16.886600   6.385370 -3.501795  0.865078 -0.788679 -0.207964  2.895361
11    0.132196   5.670740  7.593964 -0.339189 -1.754547 -2.423448  1.748274
12  -10.537500   9.512690  7.718172  5.689226  5.991894  0.598983 -0.125172
13  -50.820200   8.927082  6.986084  3.283073  4.447418  2.450394 -3.112934
14   -8.513200   1.763999  5.109834 -1.408233  1.035883 -2.301114  6.074265
15  -28.769000  -5.630880 -1.378075  2.045129 -6.512921 -0.242641  0.451446
16   -5.404050  -9.985943 -3.864202 -2.316010 -0.711510 -3.434859 -4.223349
17  -17.658400  -9.433855 -7.045584 -1.799386 -8.330568 -3.010712 -7.293407
18   14.971800   2.676125 -1.344234 -2.299299  0.999803  1.494560 -4.691778
19   64.089200  12.543811  2.555395  2.245494  4.976250 -0.141671 -2.489480

       Lag_7     Lag_8     Lag_9    Lag_10
0        NaN       NaN       NaN       NaN
1        NaN       NaN       NaN       NaN
2        NaN       NaN       NaN       NaN
3        NaN       NaN       NaN       NaN
4        NaN       NaN       NaN       NaN
5        NaN       NaN       NaN       NaN
6        NaN       NaN       NaN       NaN
7   1.702393       NaN       NaN       NaN
8   3.645954  2.205893       NaN       NaN
9   4.408004 -0.365594  0.525703       NaN
10  3.330357  3.415228 -0.883780  4.134303
11  0.031863  2.956235  1.789533 -1.509402
12 -0.076477  1.646860  0.458877  2.655329
13 -3.220253 -1.874773  0.120246  6.220969
14 -0.375654 -1.772087 -4.166345 -1.135234
15  0.307970 -4.446781 -4.877276 -7.316638
```

```
16 -2.660224  4.907276 -3.198715 -1.138683
17 -1.765303 -1.450087  1.270269 -4.213599
18 -4.305335 -5.159260 -4.342382  1.460853
19 -2.825177 -9.688401 -5.017985  1.351116
```

[24]:
```python
fig, axs = plt.subplots(10)
for i in range(10):
    axs[i].scatter(true_values[i],pred_values[i])
    axs[i].xlabel = 'True Values'
    axs[i].ylabel = 'Predicted Values'
    axs[i].set_title("Lag = "+str(i+1))
    axs[i].set_aspect('equal', adjustable='box')
```



[25]:
```
# SVM - Poly Kernel
```

[26]:
```
true_values = []
pred_values = []
rmse_values = []
for lag in range(1,11):
    # forming features using lag
    y = df['ISMR'][lag:]
    X = df.iloc[:-1*lag,:]

    # intialising SVM regressor object and fitting on data
    svr = SVR(kernel ='poly', C=1.0, epsilon=0.1)
    svr.fit(X,y)

    #generating predictions
    preds = svr.predict(X)

    # calculating error to help as compare lag performance
    rmse = np.sqrt(mean_squared_error(y, preds))

    # appending values to a list for later use
    true_values.append(y)
    pred_values.append(preds)
    rmse_values.append((rmse))
```

[27]:
```
rmse_table = pd.DataFrame(list(np.array(rmse_values).
 ↪reshape((len(rmse_values),-1)).T))
rmse_table.columns = [("Lag_"+str(i)) for i in range(1,11)]
```

[28]:
```
rmse_table
```

[28]:

	Lag_1	Lag_2	Lag_3	Lag_4	Lag_5	Lag_6	Lag_7
0	32.611828	32.92766	33.057109	33.041273	33.080376	32.981059	32.938083

	Lag_8	Lag_9	Lag_10
0	33.0812	33.088109	33.141695

[29]:
```
table = pd.DataFrame()
table['True ISMR'] = df['ISMR']
```

[30]:
```
for i in range(len(pred_values)):
    pred_lag = list(pred_values[i])
    for i in range(df.shape[0]-len(pred_lag)):
        pred_lag.insert(0,np.NaN)
```

```
    table['Lag_'+str(i+1)] = pred_lag
```

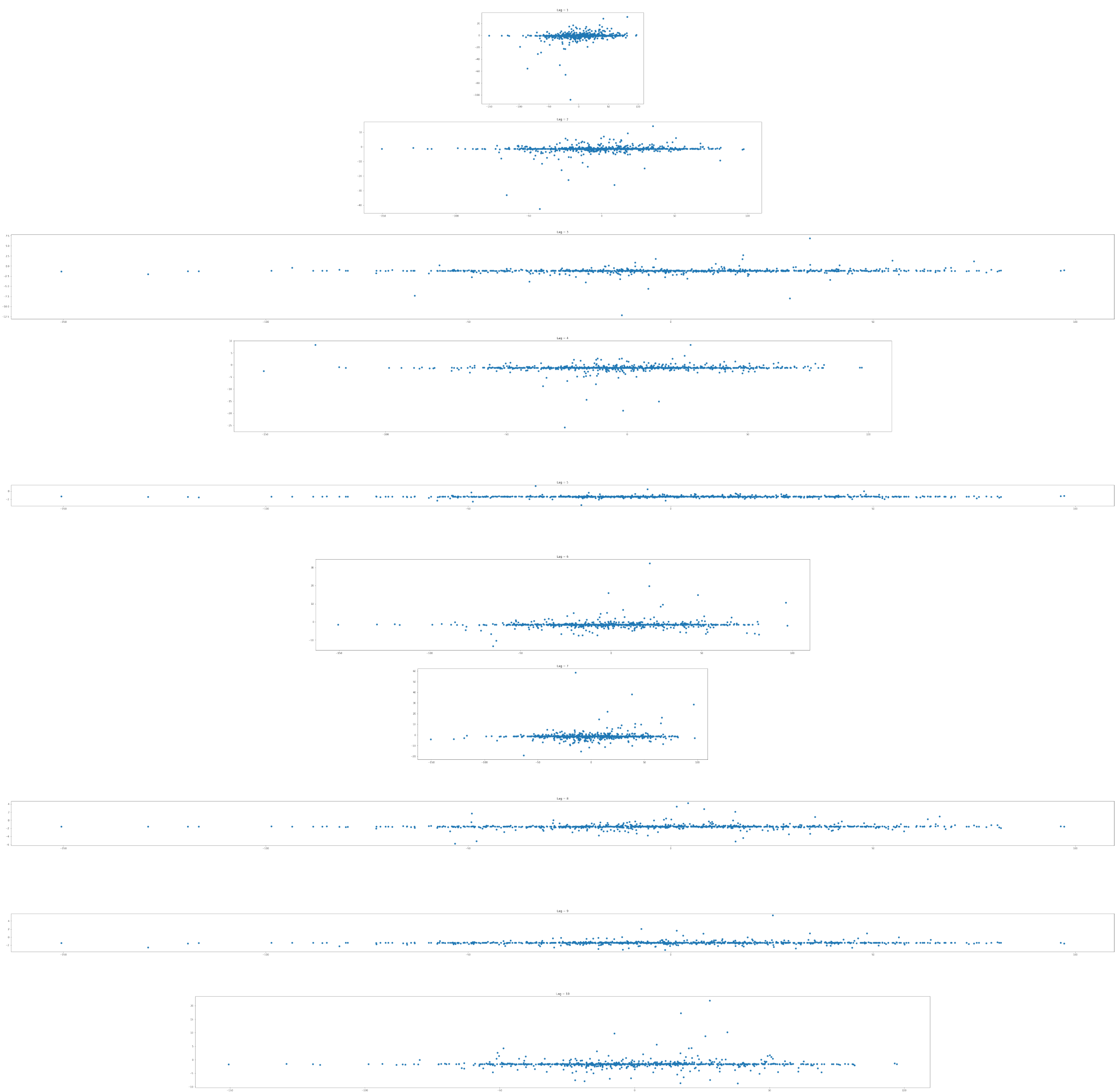
[31]:
```
table[0:20]
```

[31]:

	True ISMR	Lag_1	Lag_2	Lag_3	Lag_4	Lag_5	Lag_6
0	10.243800	NaN	NaN	NaN	NaN	NaN	NaN
1	44.161100	-0.553799	NaN	NaN	NaN	NaN	NaN
2	4.768070	1.409998	-1.337070	NaN	NaN	NaN	NaN
3	-63.587700	-0.594516	-0.480561	-1.145828	NaN	NaN	NaN
4	12.577200	-7.831394	-1.345313	-1.116867	-1.147418	NaN	NaN
5	-41.577200	-0.567826	-4.416571	-1.159616	-0.818578	-1.357415	NaN
6	-9.546880	-3.268182	-1.360824	-1.322785	-1.161940	-1.061223	-1.489415
7	25.370500	-0.600268	-2.095118	-1.140791	-2.427974	-1.350311	-1.729758
8	18.277400	-0.185252	-1.334301	-1.457471	-1.146612	-1.896288	-1.498301
9	2.221670	-0.424751	-1.026527	-1.151829	-1.736113	-1.343231	1.875892
10	16.886600	-0.594682	-1.342886	-1.237701	-1.156892	-1.219041	-1.478619
11	0.132196	-0.451335	-1.348417	-1.052843	-1.133828	-1.365148	-0.743560
12	-10.537500	-0.596784	-1.318907	-1.160308	-1.054258	-1.285997	-1.492600
13	-50.820200	-0.601772	-1.348542	-1.159339	-1.164988	-1.365201	-1.695292
14	-8.513200	-5.034933	-1.370768	-1.161943	-1.177999	-1.351717	-1.459198
15	-28.769000	-0.615795	-2.502595	-1.148397	-1.166002	-1.356384	-1.500031
16	-5.404050	-1.097010	-1.359551	-2.252933	-1.173027	-1.350597	-1.476281
17	-17.658400	-0.606590	-1.620088	-1.155168	-2.731447	-1.363818	-1.500766
18	14.971800	-0.776900	-1.358475	-1.093639	-1.165094	-1.171150	-1.491181
19	64.089200	-0.427553	-1.355529	-1.160298	-1.166036	-1.357600	-0.138279

	Lag_7	Lag_8	Lag_9	Lag_10
0	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN
7	-1.313935	NaN	NaN	NaN
8	-2.350474	-1.534397	NaN	NaN
9	-1.273009	-2.102512	-1.378408	NaN
10	1.561306	-1.551484	-1.652281	-1.574531
11	-1.280714	0.232160	-1.369586	-2.688745
12	0.656299	-1.541392	-1.157122	-1.604177
13	-1.277262	-1.595567	-1.362201	2.580865
14	-1.435055	-1.531047	-1.092181	-1.619347
15	-1.358402	-1.784142	-1.375195	-1.800530
16	-1.274456	-1.432213	-1.448149	-1.572870
17	-1.416235	-1.553786	-1.340615	-1.989255
18	-1.272676	-1.612184	-1.371604	-1.556037
19	-1.278688	-1.555441	-1.454254	-1.601690

[32]:
```
fig, axs = plt.subplots(10)
for i in range(10):
    axs[i].scatter(true_values[i],pred_values[i])
    axs[i].xlabel = 'True Values'
    axs[i].ylabel = 'Predicted Values'
    axs[i].set_title("Lag = "+str(i+1))
    axs[i].set_aspect('equal', adjustable='box')
```



[33]:
```
# SVM - Sigmoid Kernel
```

[34]:
```
true_values = []
pred_values = []
rmse_values = []
for lag in range(1,11):
```

```
    # forming features using lag
    y = df['ISMR'][lag:]
    X = df.iloc[:-1*lag,:]

    # intialising SVM regressor object and fitting on data
    svr = SVR(kernel ='sigmoid', C=1.0, epsilon=0.1)
    svr.fit(X,y)

    #generating predictions
    preds = svr.predict(X)

    # calculating error to help as compare lag performance
    rmse = np.sqrt(mean_squared_error(y, preds))

    # appending values to a list for later use
    true_values.append(y)
    pred_values.append(preds)
    rmse_values.append((rmse))
```

[35]:
```
rmse_table = pd.DataFrame(list(np.array(rmse_values).
 ↪reshape((len(rmse_values),-1)).T))
rmse_table.columns = [("Lag_"+str(i)) for i in range(1,11)]
```

[36]:
```
rmse_table
```

[36]:
```
       Lag_1      Lag_2      Lag_3      Lag_4      Lag_5      Lag_6  \
0  48.170946  43.842853  47.899698  44.426647  44.279512  45.048021

       Lag_7     Lag_8      Lag_9   Lag_10
0  47.645228  44.34741  46.269744  44.8154
```

[37]:
```
table = pd.DataFrame()
table['True ISMR'] = df['ISMR']
```

[38]:
```
for i in range(len(pred_values)):
    pred_lag = list(pred_values[i])
    for i in range(df.shape[0]-len(pred_lag)):
        pred_lag.insert(0,np.NaN)
    table['Lag_'+str(i+1)] = pred_lag
```

[39]:
```
table[0:20]
```

[39]:
```
   True ISMR      Lag_1      Lag_2      Lag_3      Lag_4      Lag_5  \
0  10.243800        NaN        NaN        NaN        NaN        NaN
1  44.161100  34.027572        NaN        NaN        NaN        NaN
2   4.768070 -12.875960  28.951401        NaN        NaN        NaN
3 -63.587700  21.048718 -13.407772  32.951564        NaN        NaN
```

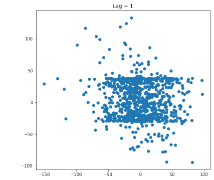
```
4    12.577200  60.093893  17.356594 -18.257679  28.537653        NaN
5   -41.577200  36.731199  51.752396  20.128548 -16.188800  30.483062
6    -9.546880  15.293926  31.329154  66.722448  16.788553 -15.400557
7    25.370500 -24.471651  12.404151  35.520899  54.457499  18.459257
8    18.277400  28.329901 -23.661365  18.679901  30.951597  57.221047
9     2.221670  37.118598  23.368752 -25.000500  13.421923  32.912049
10   16.886600  12.498256  31.451549  25.803351 -24.815825  14.904802
11    0.132196  37.820835   9.644611  35.455205  22.333600 -24.343484
12  -10.537500   4.794311  32.161871  11.677827  30.951085  23.945288
13  -50.820200 -26.119343   2.681259  36.336673   8.950194  32.791229
14   -8.513200  35.753004 -25.077095   3.967644  31.647107  10.361638
15  -28.769000 -22.582378  30.467775 -26.550503   1.816906  33.572105
16   -5.404050 -13.548241 -21.960223  40.638309 -26.250825   3.089745
17  -17.658400 -14.740350 -13.288244 -23.060209  32.257567 -25.817090
18   14.971800 -29.133364 -14.911261 -12.087316 -23.043549  34.247210
19   64.089200  38.017523 -27.549879 -15.320347 -13.570283 -22.518325

        Lag_6      Lag_7      Lag_8      Lag_9     Lag_10
0         NaN        NaN        NaN        NaN        NaN
1         NaN        NaN        NaN        NaN        NaN
2         NaN        NaN        NaN        NaN        NaN
3         NaN        NaN        NaN        NaN        NaN
4         NaN        NaN        NaN        NaN        NaN
5         NaN        NaN        NaN        NaN        NaN
6  -30.973000        NaN        NaN        NaN        NaN
7   14.127581 -30.986977        NaN        NaN        NaN
8  -19.264606  19.276019 -33.014871        NaN        NaN
9  -58.567938 -18.919056  11.075932 -29.849935        NaN
10 -33.347328 -64.835179 -20.351968  18.494421 -29.128892
11 -16.180935 -33.374085 -55.418856 -17.281655  16.914528
12  22.290031 -18.956920 -35.695479 -59.137275 -17.467653
13 -24.808506  24.208640 -13.280236 -32.439370 -58.340743
14 -33.327549 -23.321561  24.145148 -14.089381 -31.519519
15 -11.431189 -32.946625 -27.813914  27.184566 -15.376160
16 -34.029476 -10.755476 -36.094708 -23.207012  24.008558
17  -4.341066 -33.773578 -11.939305 -32.452635 -22.631951
18  23.729844  -3.396961 -36.807630  -8.903960 -31.396902
19 -35.479544  25.660374  -4.513106 -33.163886  -9.636338
```

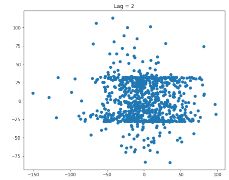
[40]:

```
fig, axs = plt.subplots(10)
for i in range(10):
    axs[i].scatter(true_values[i],pred_values[i])
    axs[i].xlabel = 'True Values'
    axs[i].ylabel = 'Predicted Values'
    axs[i].set_title("Lag = "+str(i+1))
    axs[i].set_aspect('equal', adjustable='box')
```

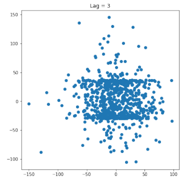
Lag = 1



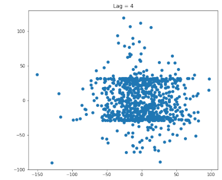
Lag = 2



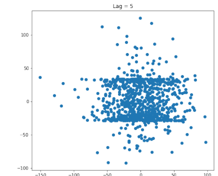
Lag = 3



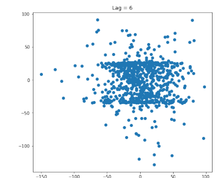
Lag = 4



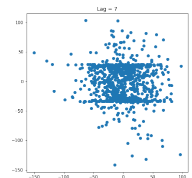
Lag = 5



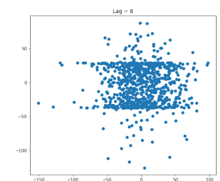
Lag = 6



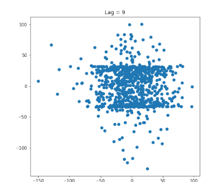
Lag = 7



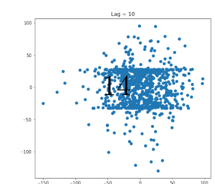
Lag = 8



Lag = 9



Lag = 10



[41]:
```
#Performing Grid Search for best kernel
```

[42]:
```
parameters = {'kernel':('linear', 'poly', 'rbf', 'sigmoid'),'C':
↪(1,2,5,10),'epsilon':(0.1,.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0,1.5,2.0)}
svr = SVR()
reg = GridSearchCV(svr, parameters)
```

[43]:
```
lag = 1

# forming features using lag
y = df['ISMR'][lag:]
X = df.iloc[:-1*lag,:]

# performing grid search
reg.fit(X,y)

#generating predictions
preds = reg.predict(X)

# calculating error to help as compare lag performance
rmse = np.sqrt(mean_squared_error(y, preds))
```

[44]:
```
rmse
```

[44]: 31.519244924329566

[45]:
```
print(reg.best_params_)
```

```
{'C': 2, 'epsilon': 2.0, 'kernel': 'linear'}
```

[46]:
```
# We use lag 1 from previous analysis. The best kernel comes out to be linear.
```

[ ]:
```

```