Forecasting Trend in the CAISO Duck Curve using Seasonal Decomposition, ARIMA and SARIMAX

UCB AIML Capstone Project

Mike Jones

12/26/2024

INTRODUCTION

Electrical Energy markets are a pervasive and a constant part of people's daily lives. We live in a society that is interwoven with electrical energy and it's applications to the point where it is present everywhere, ubiquitous, and so ordinary that people hardly notice or talk about it. Electricity is constantly at work in the many devices, products, buildings, and vehicles all around us - in and among and throughout the daily activities of people going about their lives.

Electricity just works.

Yet there is a story to be told. And, the people who work in the industry (energy, utilities, and related logistics etc) perform a continual balancing and rebalancing act in order to meet the needs of consumers throughout society.

Over the past decade, both the advent of Smart Grids, in combination with the innovation and scale of new ways to produce Solar and Wind energy have created new challenges for those who operate and manage Electrical Power Systems. Being able to accurately forecast the big picture trends in electrical power consumption, especially with respect to clean energy and renewable power sources is a vitally important task for planners and operators in the world's largest energy markets.

CRISP-DM

The analysis was conducted according to the CRISP-DM framework (where CRISP-DM stands for Cross Industry Standard Process for Data Mining). At a high level, CRISP-DM is a cyclical set of activities which goes from establishing an initial analytic objective and forming a basic understanding and then working toward achieving that objective by continually working through and refining the steps to eventually produce the result. The steps in the CRISP-DM Cycle are:

[1] Business Understanding
[2] Data Understanding
[3] Data Preparation
[4] Modeling
[5] Evaluation
[6] Deployment

By the nature of this particular assignment the steps [1] and [6] Business Understanding and Deployment (i.e. turning in the project) were established at the beginning and end of the project. The main concentration of effort was many multiple passes through the steps [2] - [5] of the cycle (Data Understanding, Preparation, Modeling, and Evaluation).

Business Understanding

There are nine ISOs (Independent System Operators) in North America, of which CAISO is one. CAISO stands for the California Independent System Operator, and is the organization that provides electrical power to most of California as well as parts of Nevada. It is comprised of four utility companies serving the region, they are:

PGE Pacific Gas and Electric
SCE Southern California Edison
SDGE San Diego Gas and Electric
VAE Valley Electric Association

This study uses the timeseries methods from Module 10 of the UCB AIML Course in order to understand and make forecasts with this data set. Specifically the "Duck Curve" is calculated, and then it's trend is determined using Seasonal Decomposition. ARIMA and SARIMAX are then used on the trend to produce 7-day forecasts for both The Total Caiso Load and The Duck Curve are likely to be across a seven day window.

Additionally, this work builds on the Machine Learning concept and principle of using Grid Search for the purpose of finding the optimal result of a given model which is run across many input permutations of a given hyperparameter search space. After a brief scan of the internet, it was found that SKLearn does not have a built in facility for conducting Grid Search on ARIMA per se. As such, that methodology was "built by hand" in the cells of the notebook below.

The work represents an initial foray, and early steps toward making these 7-day forecasts. Much was discovered and learned in the process of producing these preliminary results (both in the sense of machine learning as well as human learning). The work forms a foundation for ongoing research into this data set. It represents a basis for further development as well as potential business and industrial applications of the methods and findings.

Background information can be found in the following articles:

https://aurorasolar.com/blog/the-duck-curve-a-review-of-californias-daily-load-predictions/
Understanding the California Duck Curve for Daily Load Projections

https://www.greentechmedia.com/articles/read/eia-charts-californias-real-and-growing-duck-curve
EIA Data Reveals California's Real and Growing Duck Curve

Additionally, Wikipedia has excellent introductory articles on Smart Grids and The Duck Curve and Electrical Systems in general:

https://en.wikipedia.org/wiki/Smart_grid
https://en.wikipedia.org/wiki/Duck_curve
https://en.wikipedia.org/wiki/Electric_power_transmission

Data Understanding and Preparation

The data for the capstone was obtained from the website / firm GridStatus.io back in October 2024. It consists of two data sets: [1] Total CAISO Load and [2] Fuel Mix, documentation for which can be found here:

https://www.gridstatus.io/datasets/caiso_load
https://www.gridstatus.io/datasets/caiso_fuel_mix

These appear to be an evolving product offering from the Grid status.

The documentation found on the site, continues to evolve and become more detailed and comprehensive in scope. Its appearance is becoming more corporate and I suspect that the company is rapidly growing and professionalizing its services. Whereas CAISO Load and Fuel Mix were previously freely available, they are now apparently behind a paywall and require api keys in order to access.

Some aspects of the previously accessible data still remains freely available at the time of this writing. For example, for NYISO (New York Independent System Operator), the following python code:

```
import gridstatus
nyiso = gridstatus.NYISO()

gls0 = gridstatus.list_isos()
print(f' gls0 { gls0 }')
print(f' gridstatus.__version__ { gridstatus.__version__ }')

load0 = nyiso.get_load("today")
print(load0)

fuel0 = caiso.get_fuel_mix("today")
print(fuel0)
```

produces results, returning the following for list_isos():

```
gls0                                     Name     Id  Class
0                       Midcontinent ISO   miso   MISO
1                         California ISO  caiso  CAISO
2                                    PJM    pjm    PJM
3  Electric Reliability Council of Texas  ercot  Ercot
4                     Southwest Power Pool    spp    SPP
5                           New York ISO  nyiso  NYISO
6                         ISO New England  isone  ISONE
```

And, for NYISO get_load('today') returns:

```
   Name                  Time      Load    CAPITL    CENTRL   DUNWOD    GENESE    HUD VL    LONGIL    MHK VL   MILLWD
N.Y.C.    NORTH      WEST
   0   2024-12-26 00:00:00-05:00  17240.7136  1444.5430  1739.3518  655.1893  1007.5286  1176.1653  2241.7060   971.4618
348.3720  5250.8840  736.2094  1669.3024
   1   2024-12-26 00:05:00-05:00  17215.0717  1434.8036  1726.4724  645.1801  1006.7183  1194.5466  2217.9062   964.7794
353.7157  5236.6850  764.6216  1669.6428
   2   2024-12-26 00:10:00-05:00  17103.0779  1430.3510  1737.2461  672.1261  1007.2386  1176.1284  2207.2320   985.0500
361.6034  5190.7910  671.5379  1663.7734
   3   2024-12-26 00:15:00-05:00  16968.4203  1443.9225  1720.1631  654.9122  1006.0508  1153.6467  2200.9420   982.3648
356.3033  5189.7393  606.7305  1653.6451
   4   2024-12-26 00:20:00-05:00  16956.4011  1425.3980  1736.9382  662.0627  1004.7451  1162.4446  2194.0810   953.2435
344.7347  5190.6504  631.0508  1651.0521
   5   2024-12-26 00:25:00-05:00  16908.4444  1428.0290  1726.0913  656.2340   999.7396  1174.0730  2181.3780   975.3463
342.8573  5178.6255  605.7811  1640.2893
   6   2024-12-26 00:30:00-05:00  16903.0264  1425.3310  1725.5776  654.1423   994.6881  1147.2715  2185.0250   973.3925
350.5370  5173.7646  613.5656  1659.7312

   ...

   265  2024-12-26 21:55:00-05:00  19748.2583  1708.4192  2052.7760  757.1603  1198.2270  1379.1693  2623.3936
1167.7213  418.7491  5928.8710  614.9772  1898.7943
   266  2024-12-26 22:00:00-05:00  19637.9511  1709.2439  2068.3284  756.7487  1192.1921  1381.0676  2605.5188
1156.8776  411.1213  5874.6700  612.7719  1869.4108
```

267 2024-12-26 22:05:00-05:00 19526.3248 1677.2137 2040.6318 765.5911 1191.6926 1410.9977 2597.9695 1149.0791 397.2191 5825.2130 616.1946 1854.5226
268 2024-12-26 22:10:00-05:00 19309.8087 1655.7572 2013.1125 741.6625 1178.5254 1355.9497 2576.2870 1130.0137 403.3360 5825.6910 599.9683 1829.5054
269 2024-12-26 22:15:00-05:00 19252.6951 1672.3241 1976.7744 753.4548 1177.6731 1371.5371 2550.3145 1119.3281 418.2089 5785.9175 609.4955 1817.6671

showing the five minute incremental data for total load, and split apart into it's component operators. However, NYISO fuel_mix('today') is not available at this time, returning the following error:

urllib.error.HTTPError: HTTP Error 404: Not Found

That same error is what has been returned for CAISO Load and Fuel Mix for the last few weeks at this point. It seems unlikely to change without signing up for a paid subscription.

Fortunately in my initial Exploratory Data Analysis looking into the data set, I was able to collect and save nearly six years of data for both CAISO Load and Fuel Mix. The Python code I ran back in October which I used to create the files is:

```
caiso = gridstatus.CAISO()
start = pd.Timestamp("Jan 1, 2019").normalize()
end = pd.Timestamp.now().normalize()
fm1 = caiso.get_fuel_mix(start, end=end, verbose=True)
fm1.to_csv('fuel_mix_20190101_20241003.csv',index=False)

lod0  = caiso.get_load('2019-10-01',end='2024-10-03')
lod0.to_csv('load_20190101_20241003.csv',index=False)
```

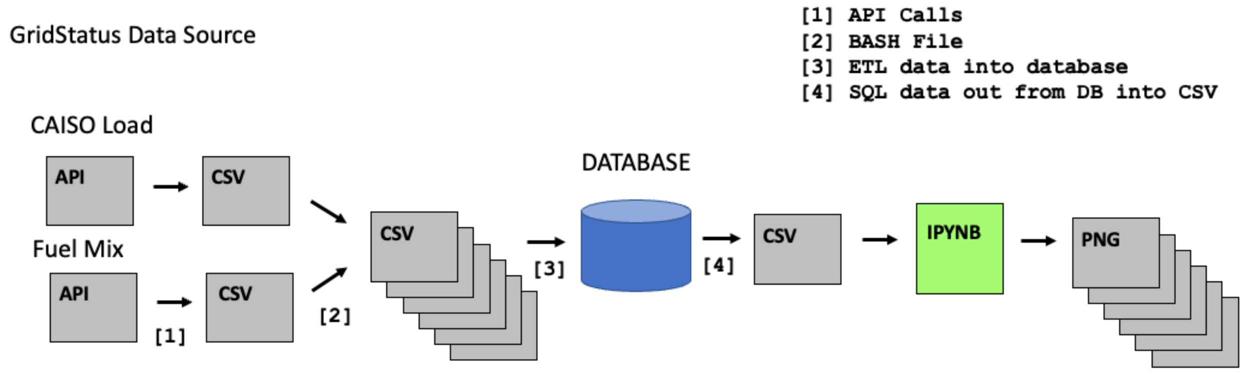
I saved the six years of data (at five minute sample intervals) aside in two CSV files, just to have it handy. The files are:

fuel_mix_20190101_20241003.csv
load_20190101_20241003.csv

The files are somewhat large. The fuel_mix file is 95.6 MB and has 605,091 records ranging from between 2019-01-01 and 2024-10-02. The load file is 45.3 MB has 526,486 records ranging from between 2019-10-01 and 2024-10-02. Working with files of this magnitude in combination was too much and overwhelmed the Jupyter Notebook.

For this project I used my laptop and did all of the work locally. When I ran into the "Big Data like" issues, I decided to do some preprocessing outside of the notebook in order to reduce the data size to something more manageable. This is depicted in the following diagram:

Data Flow Diagram - preparing the CAISO Data Set



See the files in the directory "_other" for the BASH, ETL and SQL scripts used to reduce the total data size to something more manageable. Aspects of the Feature Engineering and Data Preparation were done in the database, including: filling in missing timestamps, joining the Load and Fuel_Mix tables, adding together Solar and Wind and subtracting both from the total load to calculate The Duck Curve:

Duck = Load - (Solar + Wind)

The results were saved to a CSV so that it could be loaded up to GitHub along with the Jupyter Notebook so as to form a consistent whole. As such, the code and the data could be distributed in such a way where nothing other than the Notebook and the CSV data was needed in order to duplicate and perform the calculations. That file is:

ucb_aiml_capstone_caiso.csv

and holds data in the following format, where the last full day of data looks like:

ts0	dt0	hr0	solar	wind	caiso_load	sol_wind	duck
2024-10-02 00:00	10/2/24	0	-56	917	26333	861	25472
2024-10-02 01:00	10/2/24	1	-59	1059	25552	1000	24552
2024-10-02 02:00	10/2/24	2	-60	1152	24259	1092	23167
2024-10-02 03:00	10/2/24	3	-59	1101	23123	1042	22081
2024-10-02 04:00	10/2/24	4	-59	1085	22757	1026	21731
2024-10-02 05:00	10/2/24	5	-61	1140	23152	1079	22073
2024-10-02 06:00	10/2/24	6	-60	1189	24497	1129	23368
2024-10-02 07:00	10/2/24	7	269	1245	25889	1514	24375
2024-10-02 08:00	10/2/24	8	7712	1044	26449	8756	17693
2024-10-02 09:00	10/2/24	9	14825	969	26858	15794	11064
2024-10-02 10:00	10/2/24	10	16580	906	26753	17486	9267
2024-10-02 11:00	10/2/24	11	16444	844	27269	17288	9981
2024-10-02 12:00	10/2/24	12	16695	801	28470	17496	10974
2024-10-02 13:00	10/2/24	13	16675	801	31104	17476	13628
2024-10-02 14:00	10/2/24	14	16467	824	33908	17291	16617
2024-10-02 15:00	10/2/24	15	16224	1033	37133	17257	19876
2024-10-02 16:00	10/2/24	16	15032	1619	39782	16651	23131
2024-10-02 17:00	10/2/24	17	10912	1740	40997	12652	28345
2024-10-02 18:00	10/2/24	18	1927	1877	41185	3804	37381

2024-10-02 19:00	10/2/24	19	-48	2295	39724	2247	37477
2024-10-02 20:00	10/2/24	20	-37	2407	37673	2370	35303
2024-10-02 21:00	10/2/24	21	-29	2451	35253	2422	32831
2024-10-02 22:00	10/2/24	22	-31	2727	33028	2696	30332
2024-10-02 23:00	10/2/24	23	-32	2772	30162	2740	27422

Modeling and Evaluation

The plan for modeling is to run:

Seasonal Decomposition
ARIMA
SARIMAX

in that order.

Use the Seasonal Decomposition method to establish the component trend, seasonality, and residue from the original timeseries given by the equation:

$y = t * s * r$ (multiplicative) , or
$y = t + s + r$ (additive)

where:

y is the original timeseries,
t is the trend,
s is the seasonal component, and
r is the residue.

note that seasonality can be determined by additive or multiplicative combination of the components, and either can be used. Additive is used in the current IPYNB file, however both could be tested and included in future hyperparameter gridsearch testing.

The Seasonal Decomposition Trend was determined for both CAISO Load and Duck Curve. Once the trend was isolated, it was then used for further analysis and forecasting by ARIMA and SARIMAX. Grid search was used for determining optimal ARIMA hyper parameters, however SARIMAX forecasts did not use gridsearch.

Grid Search does not appear to be immediately and directly available for SKLearn's ARIMA and SARIMAX as indicated by the following google search result:

While scikit-learn's GridSearchCV doesn't directly support ARIMA models, you can still use it with a bit of customization.

As such, in the absence of a built in grid search method, I decided to create my own model scoring mechanism for the purpose of optimizing hyperparameters. I performed my DIY GridSearch on ARIMA over three hyperparameters:

In essence I created a triply nested for loop to search through many combinations of:

[d1] arima order,
[d2] arima period, and
[d3] forecasts date range (7 day window starting on diff days/date).

For any one arima_order and one arima_period, several 7-day forecasts are run and the accuracy of the forecasts are scored with:

Mean Square Error (MSE),
Root Mean Square Error (RMSE),
Mean Absolute Error (MAE), and
Mean Absolute Percentage Error (MAPE).

This triply nested for loop was used to produce 100 different model runs for both Caiso Load and Duck Curve. They are based on the first two dimensions [d1] & [d2] form a 10x10 cross product / cartisian grid denoted as X_Y where X is the arima_order, and Y is the arima_period. I give the column containing these values the name "IDX" for "index" in the result files as well as the excel file mentioned below. Then, once all of the gridsearch runs with the different hyperparameters are completed their result is collected are written out to the files:

result_caiso_load.txt , and
result_duck.txt

and then put into the following excel file for further analysis:

rpt_0_caiso_load_duck_matrix.xlsx

The file has three sheets:

caiso_scores
duck_scores
matrix

where the caiso_scores and duck_scores sheets correspond to the result_caiso_load.txt and result_duck.txt files.

The results can then be cross compared and evaluated by looking at which runs have lower or higher values for MSE, RMSE, MAE, and MAPE. The IDX's with lower scores indicate the most accurate forecasts, and the IDX's with higher scores indicate the least accurate forecasts. Scores for MSE, RMSE, MAE and MAPE are colored in from blue to red or green to red to show how that forecast performed.

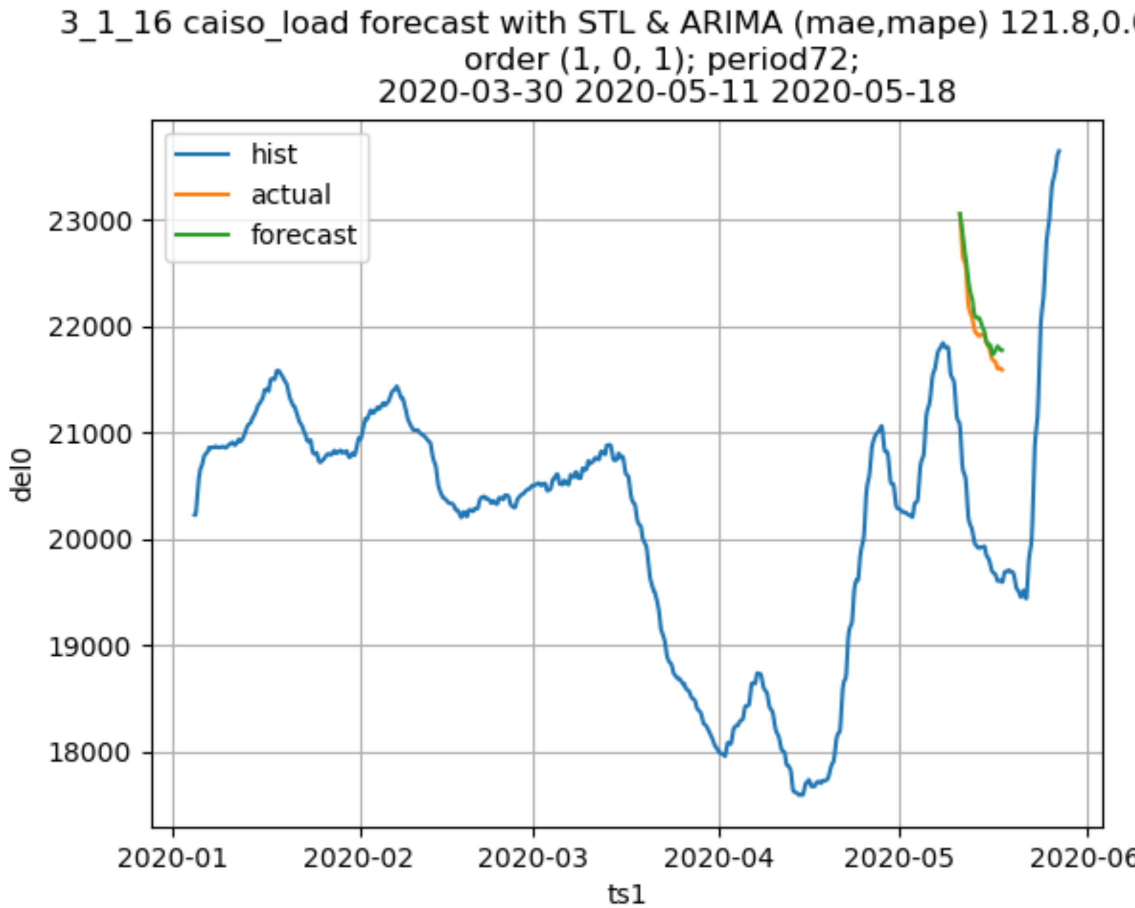
Strongly blue or green cells indicate the best relative scores in the column and strongly red cells indicates the least accurate hyper parameters. The matrix shows a condensed picture of the best and worst scores for MAPE, with worst and best identified:

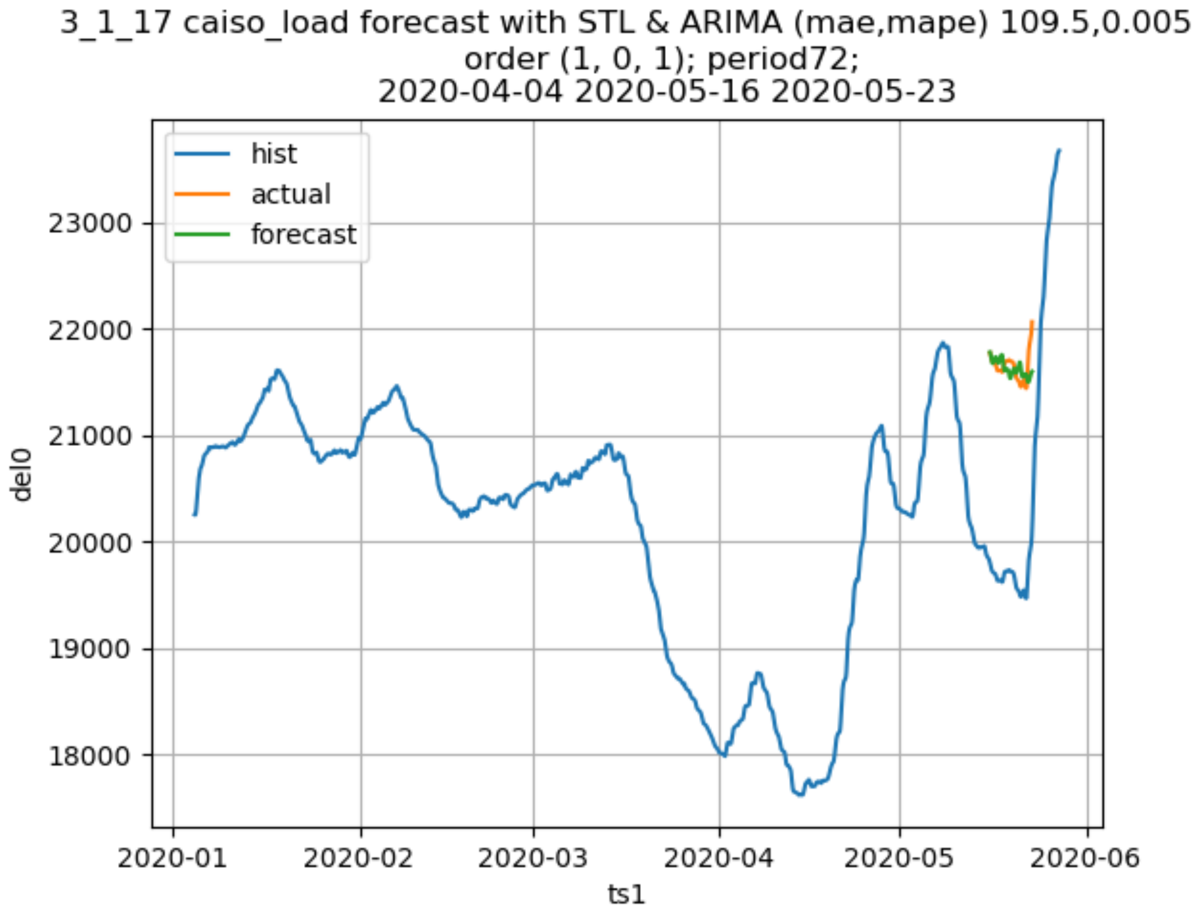
	worst	best
duck	0.104	0.054
load	0.051	0.025

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**Optimization Matrices using MAPE**												
2													
3			24	72	120	168	216	264	312	360	408	456	<<< arima_period
4		ord1	mape3	mape3	mape3	mape3	mape3	mape3	mape3	mape3	mape3	mape3	
5	duck	(0,0,1)	0.058	0.056	0.061	0.067	0.069	0.071	0.075	0.072	0.072	0.067	
6		(0,1,0)	0.089	0.083	0.088	0.083	0.086	0.087	0.102	0.092	0.084	0.081	
7		(0,1,1)	0.088	0.082	0.087	0.088	0.087	0.088	0.102	0.093	0.084	0.080	
8		(1,0,0)	0.065	0.056	0.055	0.063	0.066	0.073	0.079	0.075	0.075	0.068	
9		(1,0,1)	0.059	0.054	0.055	0.063	0.067	0.072	0.077	0.073	0.073	0.066	
10		(1,1,0)	0.089	0.084	0.083	0.091	0.087	0.088	0.104	0.091	0.080	0.078	
11		(1,1,1)	0.089	0.082	0.086	0.093	0.087	0.089	0.102	0.093	0.083	0.080	
12		(1,1,2)	0.083	0.082	0.086	0.092	0.088	0.090	0.101	0.095	0.083	0.078	
13		(2,1,1)	0.089	0.082	0.087	0.091	0.087	0.089	0.102	0.091	0.087	0.085	
14		(2,1,2)	0.074	0.074	0.069	0.076	0.081	0.089	0.100	0.085	0.081	0.082	
15													
16													
17	load	(0,0,1)	0.032	0.026	0.028	0.030	0.032	0.032	0.036	0.033	0.031	0.033	
18		(0,1,0)	0.039	0.033	0.042	0.041	0.037	0.037	0.039	0.044	0.037	0.036	
19		(0,1,1)	0.026	0.026	0.027	0.028	0.030	0.032	0.033	0.030	0.031	0.032	
20		(1,0,0)	0.029	0.025	0.027	0.029	0.031	0.031	0.035	0.031	0.031	0.033	
21		(1,0,1)	0.038	0.039	0.051	0.051	0.040	0.039	0.039	0.046	0.034	0.034	
22		(1,1,0)	0.039	0.033	0.042	0.043	0.037	0.037	0.039	0.044	0.037	0.036	
23		(1,1,1)	0.038	0.035	0.044	0.048	0.037	0.038	0.038	0.045	0.035	0.035	
24		(1,1,2)	0.038	0.034	0.041	0.042	0.034	0.038	0.038	0.038	0.034	0.033	
25		(2,1,1)	0.038	0.034	0.043	0.048	0.037	0.038	0.038	0.045	0.034	0.033	
26		(2,1,2)	0.032	0.030	0.034	0.039	0.035	0.031	0.038	0.037	0.034	0.033	
27													

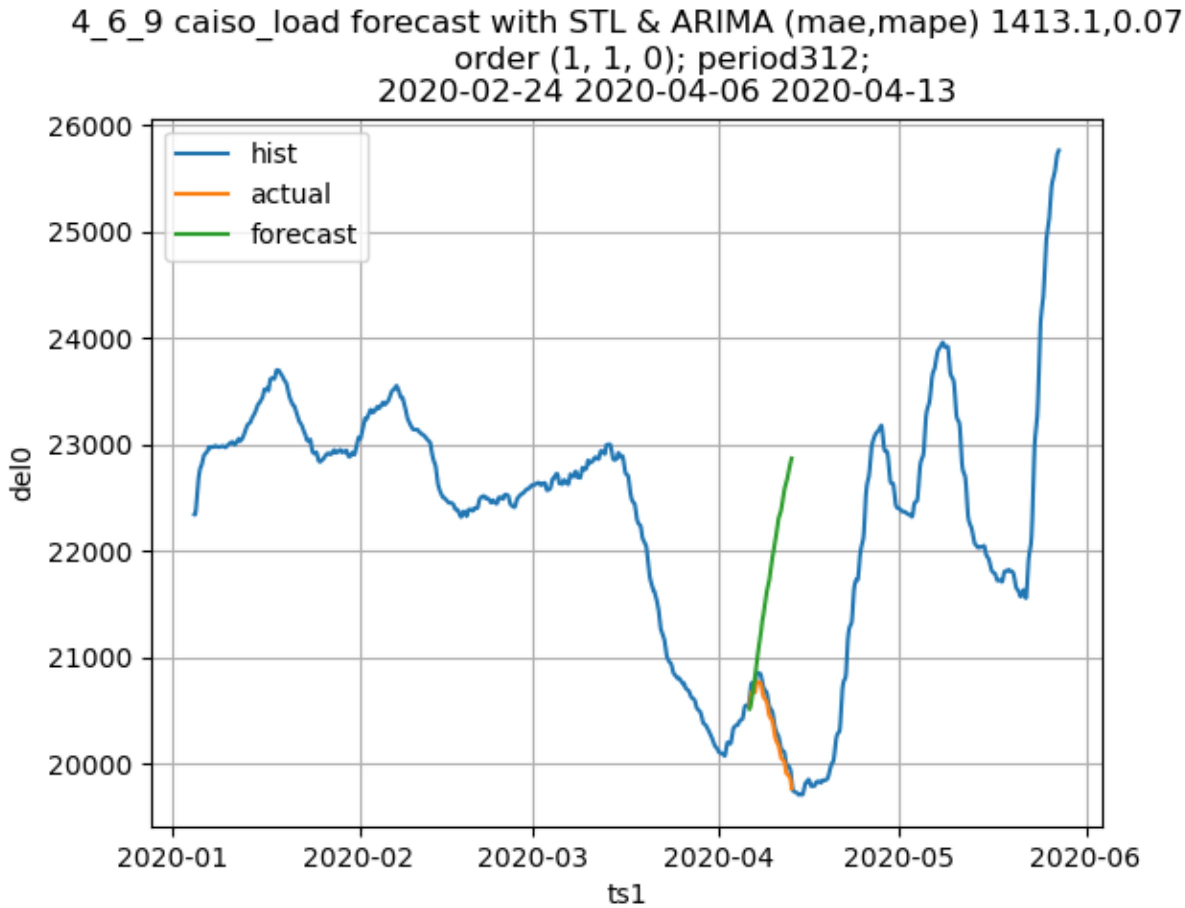
Both good and bad examples can be drawn from the best and worst performing forecasts as determined by the gridsearch hyperparameter optimization.

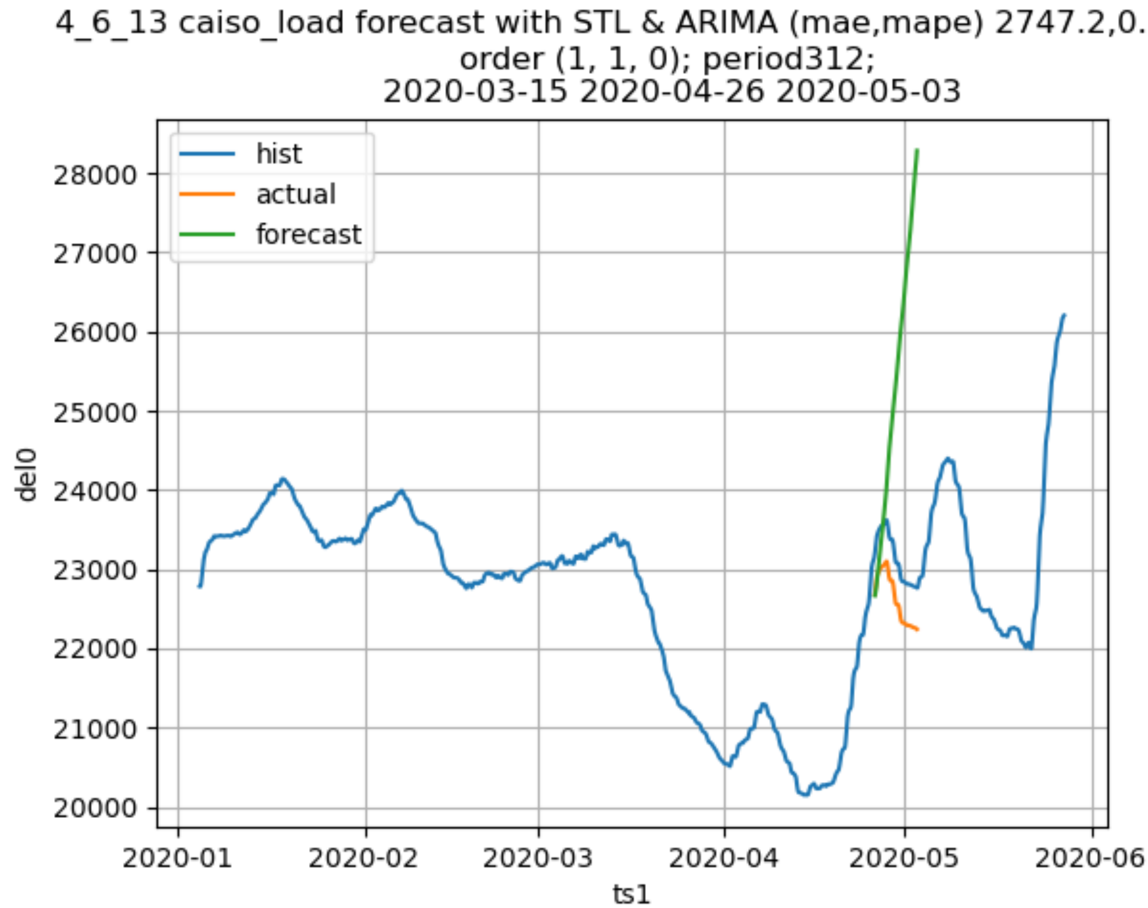
The best Duck Curve forecasts have relatively low MAE and MAPE scores, and the forecast / actual segments appear to overlap each other, one on top of another, as in the following plot.



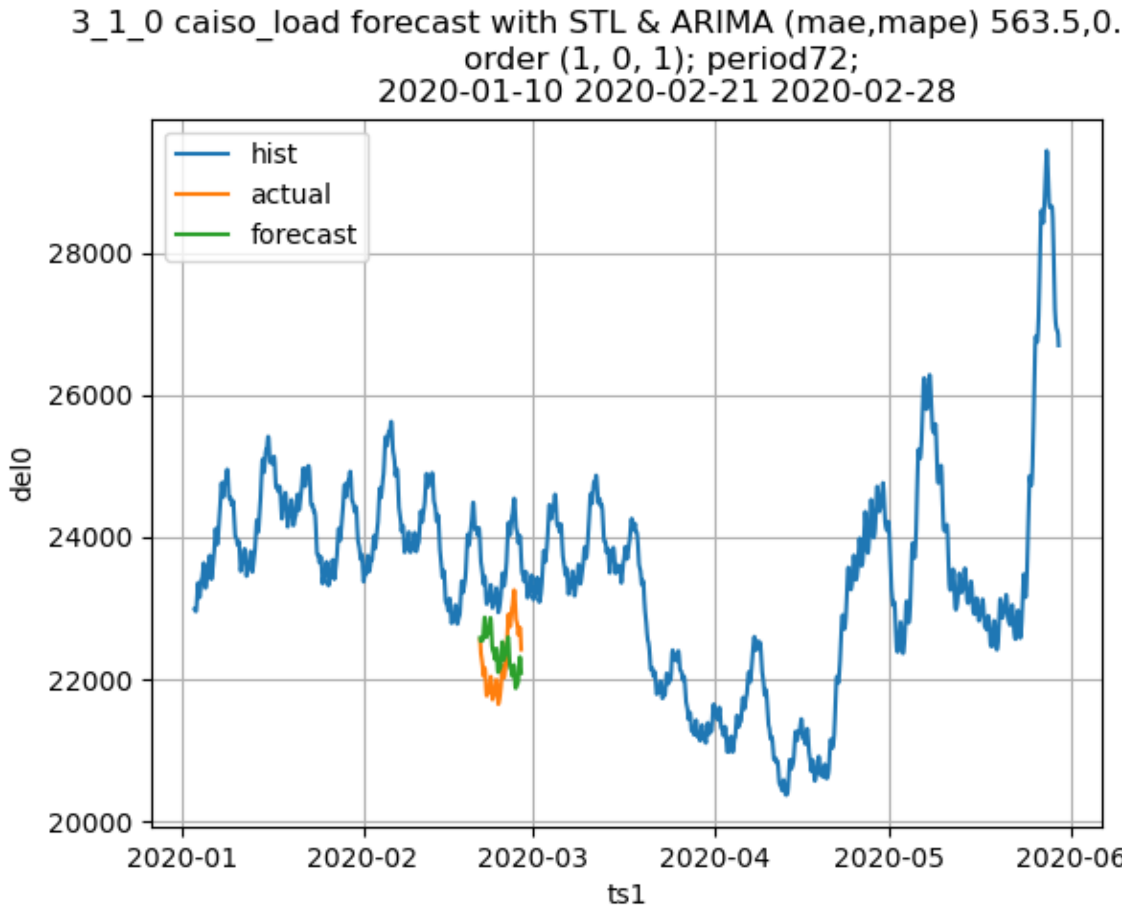


And the worst performing forecasts have relatively high MAE and MAPE scores for the forecast accuracy.


4_6_9 caiso_load forecast with STL & ARIMA (mae,mape) 1413.1,0.07
order (1, 1, 0); period312;
2020-02-24 2020-04-06 2020-04-13
hist
actual
forecast
del0
ts1
26000
25000
24000
23000
22000
21000
20000
2020-01
2020-02
2020-03
2020-04
2020-05
2020-06

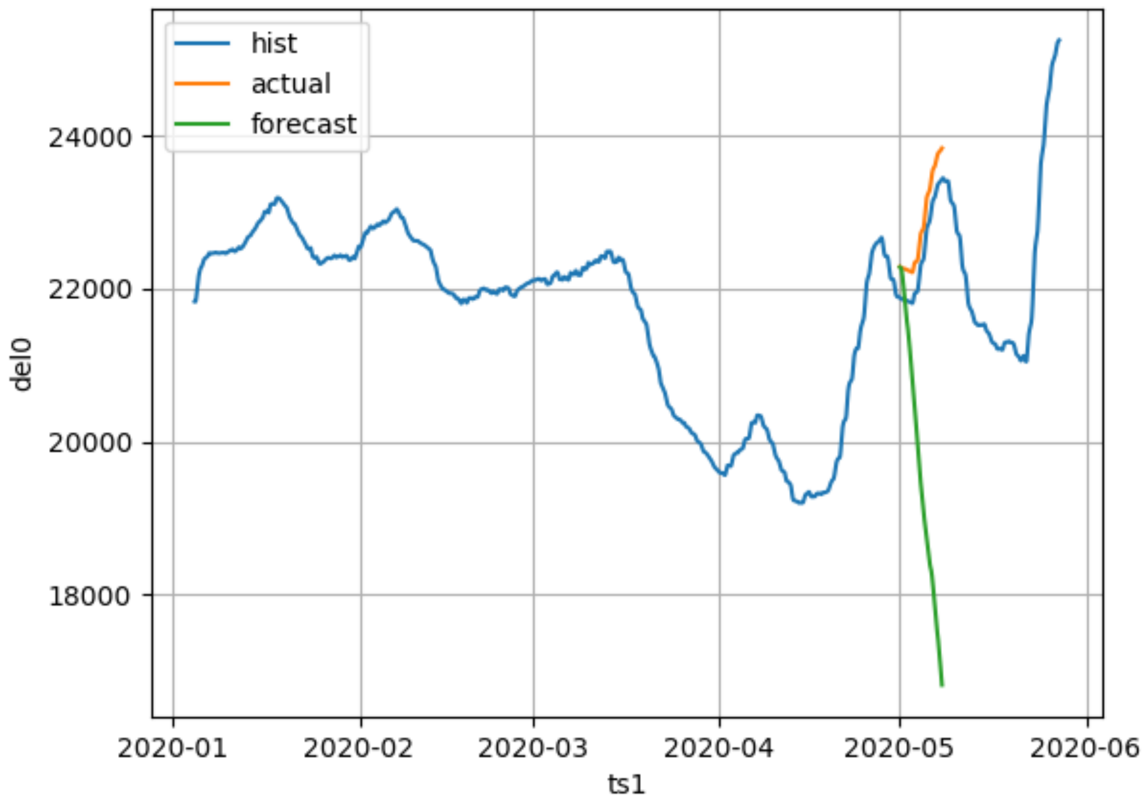


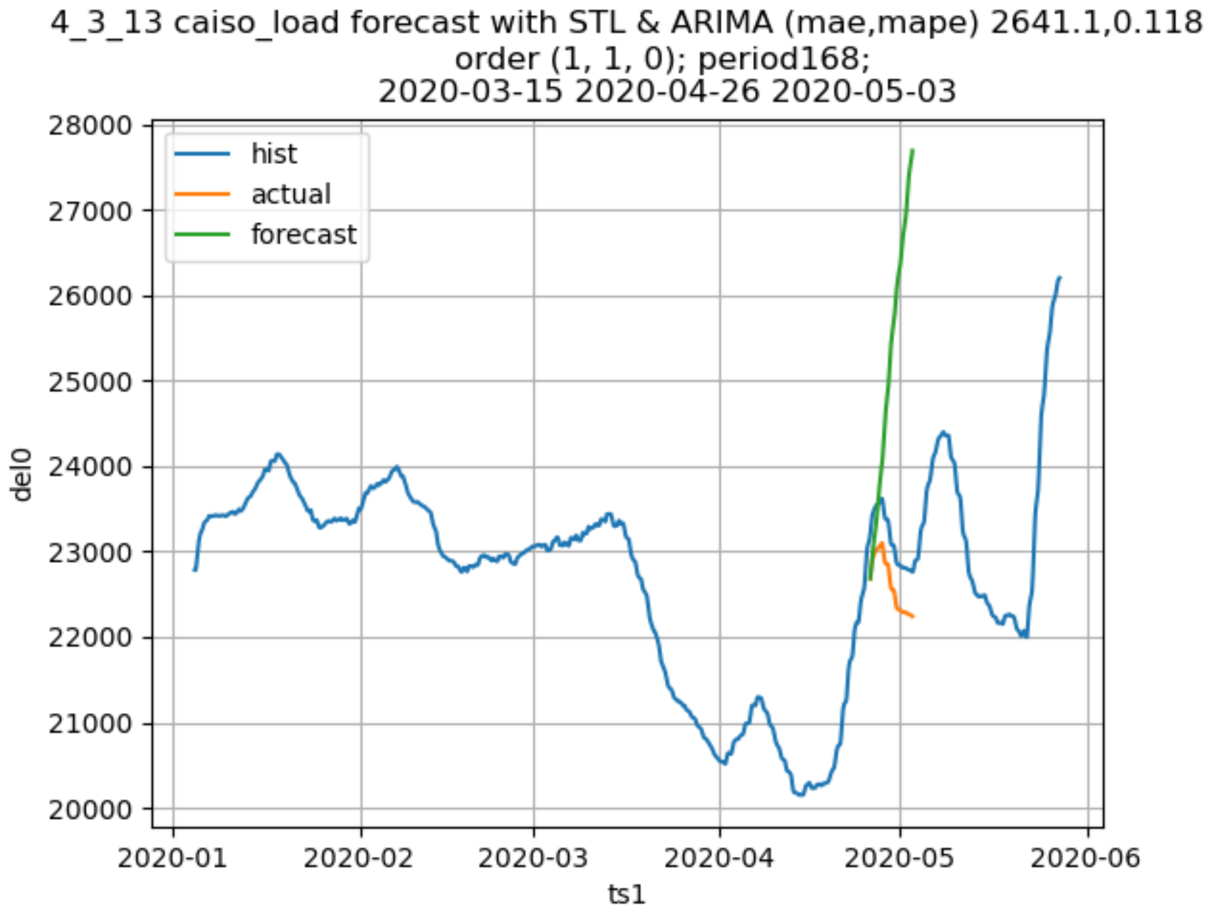
The best CAISO LOAD forecasts have relatively low MAE and MAPE scores, and the forecast / actual segments appear to overlap each other, one on top of another, as in the following plot.



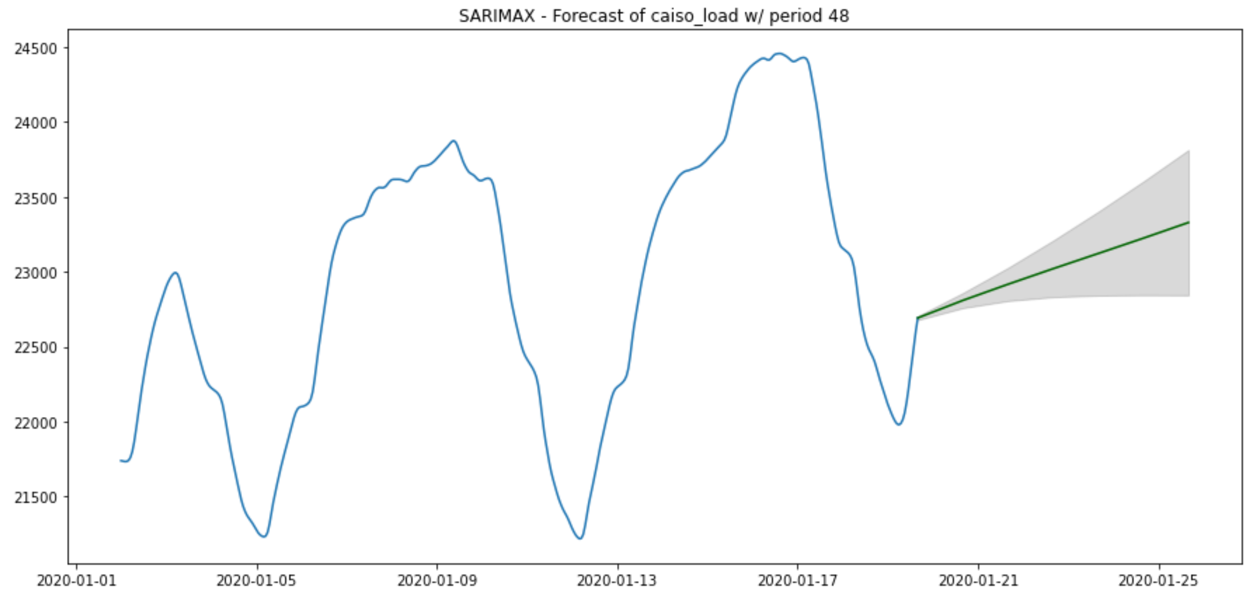
The worst CAISO LOAD forecasts have a relatively high MAE / MAPE and the forecast vs actual segments appear to diverge.

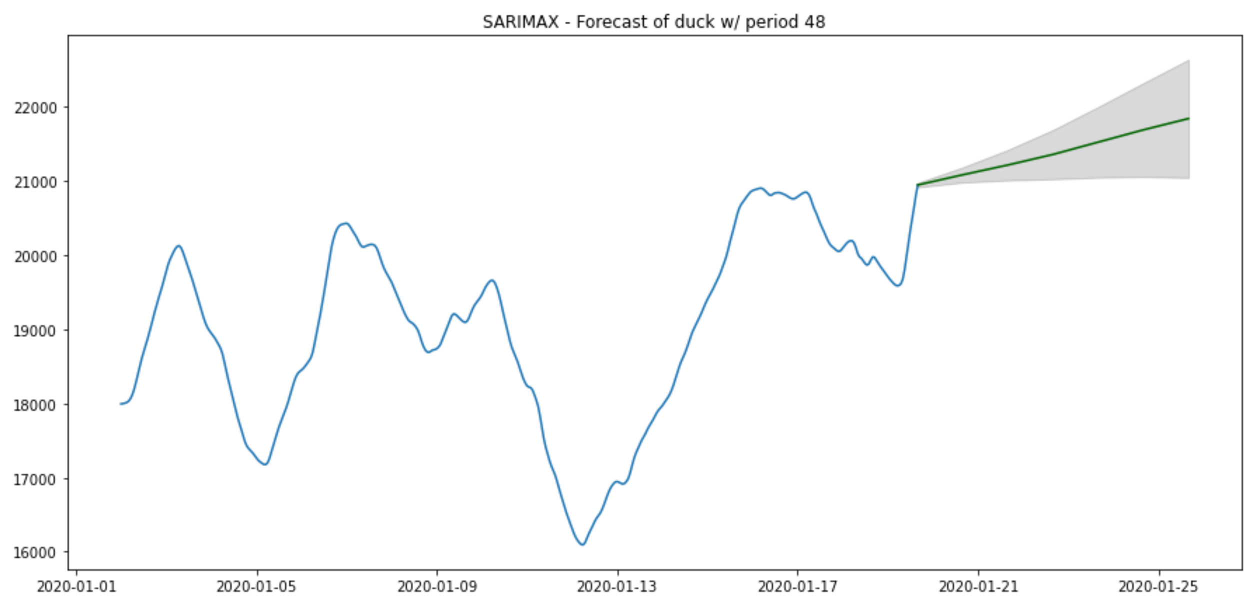
4_2_14 caiso_load forecast with STL & ARIMA (mae,mape) 3238.1,0.139
order (1, 1, 0); period120;
2020-03-20 2020-05-01 2020-05-08





These SARIMAX Forecasts for Total CAISO Load and Duck Curve predict a continuation of the average with a slight upward trend.





Deployment

In the context of academic schoolwork, the deployment is simply to upload and submit the assignment and it's findings to GitHub for grading.

In a industrial / commercial setting the result could be deployed into production operations and run over and over again for continual and constant updating of the sensor data and thereby keeping the near real time data as up-to-date as possible. It is possible to envision an alerting or monitoring system that would be based on this data and/or analysis.

Additionally there would be a component to socialize the results, by communicating the findings and making recommendations for follow on actions to decision makers, equipment operators, line workers & so forth.

CONCLUSION

One procedure has now been shown for creating many CAISO Load and Duck Curve forecasts using Seasonal Decomposition Trend, ARIMA and SARIMAX. By varying the model hyperparamenters with Grid Search, best and worst performing forecasts can be discovered in the hyperparameter search space. These findings can then be applied in a commercial / industrial setting to make operations more efficient and effective.