Analysis of Electricity Demand, Consumption, and Electric Vehicle Trends Across Regions from 2016 to 2021

Processing Big Data for Analytics Applications
Fall 2023

Cleaning

```
scala> :load data_cleaning.scala
Loading data_cleaning.scala...
import org.apache.spark.sql.{SparkSession, DataFrame}
import org.apache.spark.sql.functions._
import org.apache.spark.sql.types._
import org.apache.spark.sql.types.DoubleType
cleanAndConsolidate: (filePath: String)org.apache.spark.sql.DataFrame
23/12/03 16:36:29 WARN org.apache.spark.sql.SparkSession$Builder: Using an existing SparkSession; some spark core configurations may not take effect.
data2016a: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
23/12/03 16:36:41 WARN org.apache.spark.sql.SparkSession$Builder: Using an existing SparkSession; some spark core configurations may not take effect.
data2016b: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
23/12/03 16:36:42 WARN org.apache.spark.sql.SparkSession$Builder: Using an existing SparkSession; some spark core configurations may not take effect.
data2017a: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
23/12/03 16:36:43 WARN org.apache.spark.sql.SparkSession$Builder: Using an existing SparkSession; some spark core configurations may not take effect.
data2017b: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
23/12/03 16:36:44 WARN org.apache.spark.sql.SparkSession$Builder: Using an existing SparkSession; some spark core configurations may not take effect.
data2018a: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
23/12/03 16:36:45 WARN org.apache.spark.sql.SparkSession$Builder: Using an existing SparkSession; some spark core configurations may not take effect.
data2018b: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
23/12/03 16:36:46 WARN org.apache.spark.sql.SparkSession$Builder: Using an existing SparkSession; some spark core configurations may not take effect.
data2019a: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
23/12/03 16:36:47 WARN org.apache.spark.sql.SparkSession$Builder: Using an existing SparkSession; some spark core configurations may not take effect.
data2019b: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
23/12/03 16:36:48 WARN org.apache.spark.sql.SparkSession$Builder: Using an existing SparkSession; some spark core configurations may not take effect.
data2020a: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
23/12/03 16:36:49 WARN org.apache.spark.sql.SparkSession$Builder: Using an existing SparkSession; some spark core configurations may not take effect.
data2020b: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
23/12/03 16:36:50 WARN org.apache.spark.sql.SparkSession$Builder: Using an existing SparkSession; some spark core configurations may not take effect.
data2021a: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
23/12/03 16:36:51 WARN org.apache.spark.sql.SparkSession$Builder: Using an existing SparkSession; some spark core configurations may not take effect.
data2021b: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
finalResult: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Balancing Authority: string, Data Date: string ... 10 more fields]
header: Seq[String] = List(Balancing Authority, Data Date, Year, MonthYear, Hour Number, Local Date, Local Time, Demand Forecast (MW), Demand (MW), Net Generation (MW), Region, For
ecast Higher Than Demand)
finalResultWithHeader: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]

scala>
```

Here in this code, a function named CleanAndConsolidate has been created to clean all 12 datasets from the period of 2016 to 2021 (every 6 months).
Cleaning processes that are done:

- Data fields inside the csv file contained "," so had read as csv and removed the "," within each field
- Initially, the dataset was (3422640, 42) removed the unwanted columns
- Separated the local time into date and time
- Created binary column based on Demand Forecast and Demand

Finally, I merged all the datasets and created a common header for the final dataset

Profiling

Final data Schema

```
data: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
cleandf: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
root
 |-- Balancing Authority: string (nullable = true)
 |-- Data Date: string (nullable = true)
 |-- Year: integer (nullable = true)
 |-- MonthYear: string (nullable = true)
 |-- Hour Number: integer (nullable = true)
 |-- Local Date: string (nullable = true)
 |-- Local Time: string (nullable = true)
 |-- Demand Forecast (MW): double (nullable = true)
 |-- Demand (MW): double (nullable = true)
 |-- Net Generation (MW): double (nullable = true)
 |-- Region: string (nullable = true)
 |-- Forecast Higher Than Demand: integer (nullable = true)
```

The shape of the dataset

```
numRows: Long = 3422640
numCols: Int = 12
Number of rows: 3422640
Number of columns: 12
```

The number of empty rows in the dataset

```
emptyValuesCount: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
res3: org.apache.spark.sql.DataFrame = [Empty_Demand_Forecast: bigint, Empty_Demand: bigint ... 1 more field]
+---------------------+------------+--------------------+
|Empty_Demand_Forecast|Empty_Demand|Empty_Net_Generation|
+---------------------+------------+--------------------+
|               563692|      569470|               57266|
+---------------------+------------+--------------------+
```

In a dataset of 3 Million rows, removing almost 1 million rows could impact the data analysis, so I looked for ways to impute data.

First, look at the linear relation between the columns with the correlation matrix. From the following, we found that there is no linear relation between the columns, but still attempting to build a linear regression model for imputing.

```
correlationMatrix: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 10 more fields]
res5: org.apache.spark.sql.DataFrame = [Corr_Demand_Forecast_Net_Generation: double, Corr_Demand_Net_Generation: double]
+-----------------------------------+--------------------------+
|Corr_Demand_Forecast_Net_Generation|Corr_Demand_Net_Generation|
+-----------------------------------+--------------------------+
|               0.005605297390471968|        0.5571405099469079|
+-----------------------------------+--------------------------+
```

Building the model for imputing values -> by splitting the datasets where there are empty values, and there are not. Since the number of null values for net generation is relatively low, we drop that and use net generation as the feature and demand as the label for the model. We then tried testing the same model on the train set itself, but the accuracy score was still very low, as expected from the correlation matrix results.

```
model: org.apache.spark.ml.regression.LinearRegressionModel = LinearRegressionModel: uid=linReg_66c8057fb318, numFeatures=1
predictions: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 12 more fields]
evaluator: org.apache.spark.ml.evaluation.RegressionEvaluator = RegressionEvaluator: uid=regEval_b4bcbd614c6e, metricName=r2, throughOrigin=false
r2: Double = 0.3104055478239208
Goodness of fit on training data: 0.3104055478239208
```

So impute values by mean imputer method.

```
columnsToImpute: Array[String] = Array(Demand Forecast (MW), Demand (MW), Net Generation (MW))
imputer: org.apache.spark.ml.feature.Imputer = imputer_a5a9b2940df6
imputedData: org.apache.spark.sql.DataFrame = [Balancing Authority: string, Data Date: string ... 13 more fields]
+-------------------+----------+----+---------+-----------+----------+----------+--------------------+-----------+-------------------+------+--------------------------+----------
-----------------+-------------------+--------------------------+
|Balancing Authority| Data Date|Year|MonthYear|Hour Number|Local Date|Local Time|Demand Forecast (MW)|Demand (MW)|Net Generation (MW)|Region|Forecast Higher Than Demand|Demand Fore
cast (MW)_imputed|Demand (MW)_imputed|Net Generation (MW)_imputed|
+-------------------+----------+----+---------+-----------+----------+----------+--------------------+-----------+-------------------+------+--------------------------+----------
-----------------+-------------------+--------------------------+
|                AEC|01/01/2016|2016|  01/2016|          1|01/01/2016|   1:00:00|               733.0|      366.0|              522.0|    SE|                         1|
           733.0|              366.0|                     522.0|
|                AEC|01/01/2016|2016|  01/2016|          2|01/01/2016|   2:00:00|               706.0|      348.0|              487.0|    SE|                         1|
           706.0|              348.0|                     487.0|
|                AEC|01/01/2016|2016|  01/2016|          3|01/01/2016|   3:00:00|               698.0|      341.0|              493.0|    SE|                         1|
           698.0|              341.0|                     493.0|
|                AEC|01/01/2016|2016|  01/2016|          4|01/01/2016|   4:00:00|               695.0|      343.0|              492.0|    SE|                         1|
           695.0|              343.0|                     492.0|
```

Analytics

Calculating all the basic stats for demand forecast, demand, and net generation

```
calculateStats: (df: org.apache.spark.sql.DataFrame, colName: String)String
statsDemandForecast: String = Column: Demand Forecast (MW)_imputed, Mean: 8297.11880349166, Median: 2842.0, Mode: 8297.118803489955, Standard Deviation: 15790.460671137822
statsDemand: String = Column: Demand (MW)_imputed, Mean: 9700.597181036786, Median: 2952.0, Mode: 9700.597181030222, Standard Deviation: 1444997.1369962797
statsNetGeneration: String = Column: Net Generation (MW)_imputed, Mean: 19182.75262036353, Median: 1448.0, Mode: 0.0, Standard Deviation: 2592637.7977732616
Column: Demand Forecast (MW)_imputed, Mean: 8297.11880349166, Median: 2842.0, Mode: 8297.118803489955, Standard Deviation: 15790.460671137822
Column: Demand (MW)_imputed, Mean: 9700.597181036786, Median: 2952.0, Mode: 9700.597181030222, Standard Deviation: 1444997.1369962797
Column: Net Generation (MW)_imputed, Mean: 19182.75262036353, Median: 1448.0, Mode: 0.0, Standard Deviation: 2592637.7977732616
```

Aggregating by year and region

Year	Region	avg(Demand Forecast (MW)_imputed)	avg(Demand (MW)_imputed)	avg(Net Generation (MW)_imputed)
2021	CAL	6073.440388303899	6159.315139747079	4630.258527612237
2020	CAL	6130.200073910873	6135.629503382906	4171.717934742876
2019	CAL	6156.518172345504	2334.0801422537606	5127.7366858595315
2018	CAL	6372.462530930527	6454.204939247496	4918.723271346054
2017	CAL	6490.630654626825	6515.634162059897	5001.804337132999
2016	CAL	6464.8930498928075	6432.983419645054	4865.95866206949
2021	CAR	5597.57697764655	5808.285419295379	4148.494855794077
2020	CAR	5455.622965005497	5637.939961021429	4038.1385568255655
2019	CAR	5671.403268576405	5822.201837440155	4326.143542968146
2018	CAR	5901.34267254899	6158.935819940604	4920.673570020897
2017	CAR	5655.460076426474	5900.619970285446	4515.4114900284685
2016	CAR	5614.53493234193	5812.18759559083	4203.058374351534
2021	CENT	15799.503473246785	15303.561358447489	15814.35462328767
2020	CENT	15586.087335755312	14951.561703096539	15440.561987704918
2019	CENT	15977.308729853754	15419.285901826484	15996.742865296803
2018	CENT	15706.105143133718	11934.07603514916	13910.658206950031
2017	CENT	14853.242302375304	14700.73864155251	15195.35553652968
2016	CENT	14472.099424605334	14760.830031876138	15199.030225409837
2021	FLA	2927.590940626607	3100.467310149645	3053.4965946207185
2020	FLA	3161.813188938977	3261.2838636221322	3335.401269645082

only showing top 20 rows

Aggregating by month/year and region

MonthYear	Region	avg(Demand Forecast (MW)_imputed)	avg(Demand (MW)_imputed)	avg(Net Generation (MW)_imputed)
12/2021	CAL	6007.574113593419	6146.040501260603	4238.342506929411
12/2020	CAL	5779.510981527892	5837.403583963632	3997.5355007765274
12/2019	CAL	5794.300537634409	5868.020967741935	3617.9279569892474
12/2018	CAL	5773.046734216065	5826.771646964189	4117.421120464899
12/2017	CAL	5967.075537634409	6087.924820522771	4397.349748088355
12/2016	CAL	6011.977688172043	6126.852508694815	4511.994640561474
11/2021	CAL	5577.896172695567	5640.623466624419	4033.0027053038452
11/2020	CAL	5639.380458765958	5613.655624606594	3991.8134724353413
11/2019	CAL	5743.4334407354445	5709.176733394674	3822.5351791261496
11/2018	CAL	5868.136353356804	5820.9778429647195	4302.7396174063215
11/2017	CAL	5891.00965918752	5951.509661408836	4714.684674495076
11/2016	CAL	6005.670097799589	5756.2766626068205	4427.510444081945
10/2021	CAL	5673.900268817204	5790.7029569892475	4483.584408602151
10/2020	CAL	6268.784139784946	6280.4239247311825	4561.179301075269
10/2019	CAL	5855.811827956989	5811.21349310808	4045.178092806646
10/2018	CAL	6013.499731182796	6047.178225806451	4534.337365591397
10/2017	CAL	6252.89059139785	6294.07876344086	4832.936559139785
10/2016	CAL	6208.910215053764	5835.532947856306	4819.908996637104
09/2021	CAL	6789.3175	6897.615833333333	5225.008888888889
09/2020	CAL	7126.8358333333335	7074.343055555555	5135.726388888889

only showing top 20 rows

Aggregating by hour and region

Hour Number	Region	avg(Demand Forecast (MW)_imputed)	avg(Demand (MW)_imputed)	avg(Net Generation (MW)_imputed)
25	CAL	7655.914188062648	7973.385633044415	9845.309294132934
24	CAL	5967.360132389102	5421.479287128092	4094.4346915082024
23	CAL	6465.698466919783	5904.049598012336	4639.165140628326
22	CAL	6973.276934073067	6480.135620424557	4835.499674821691
21	CAL	7303.756496116863	6965.3189410780315	5632.090688073582
20	CAL	7451.52638662781	6520.000875384599	5633.415596832706
19	CAL	7509.602116554818	6084.338466625475	5489.771527489641
18	CAL	7427.233229693504	6359.216714543452	5645.210282290768
17	CAL	7158.301295386936	5840.007371220553	5836.6369020554
16	CAL	6916.556131153359	6040.675126437801	5609.387061966017
15	CAL	6705.231313635111	5705.4966415507515	5514.596231195877
14	CAL	6515.135693197155	5838.731021627206	5763.551225916582
13	CAL	6346.943722394235	5383.513412138154	5271.7537580899725
12	CAL	6244.375656700805	5324.487571950481	5155.628439225075
11	CAL	6174.100109255549	5821.4592340776535	5062.908296606297
10	CAL	6120.4331384526295	5498.99003545413	4988.486194699528
9	CAL	6070.249470569416	5562.824032833316	4772.890185770597
8	CAL	5969.086788087665	5333.253540389792	4452.385646297937
7	CAL	5733.034233343139	5417.364726263973	4354.690185770596
6	CAL	5386.685875678905	5007.891149364432	3706.2456376520167

only showing top 20 rows

Net growth in the demand and net generation every year per region

demandWithGrowthRates: org.apache.spark.sql.DataFrame = [Year: int, Region: string ... 5 more fields]

Year	Region	avg(Demand Forecast (MW)_imputed)	avg(Demand (MW)_imputed)	avg(Net Generation (MW)_imputed)	demand_growth	net_growth
2016	CENT	14472.099424605334	14760.830031876138	15199.030225409837	0.0	0.0
2017	CENT	14853.242302375304	14700.73864155251	15195.35553652968	-0.4071003472965886	-0.02417712726180268
2018	CENT	15706.105143133718	11934.07603514916	13910.658206950031	-18.81988840059515	-8.454539457739127
2019	CENT	15977.308729853754	15419.285901826484	15996.742865296803	29.20385169670794	14.996304468932491
2020	CENT	15586.087335755312	14951.561703096539	15440.561987704918	-3.033371335792806	-3.4768382681105625
2021	CENT	15799.503473246785	15303.561358447489	15814.35462328767	2.354266814001438	2.420848644501401
2016	CAR	5614.53493234193	5812.18759559083	4203.058374351534	0.0	0.0
2017	CAR	5655.460076426474	5900.619970285446	4515.4114900284685	1.521498975045147	7.431567393472741
2018	CAR	5901.34267254899	6158.935819940604	4920.673570020897	4.377774724622072	8.975086343456901
2019	CAR	5671.403268576405	5822.201837440155	4326.143542968146	-5.467405284695703	-12.082289519770493
2020	CAR	5455.622965005497	5637.939961021429	4038.1385568255655	-3.1648143015897356	-6.6573146101615865
2021	CAR	5597.57697764655	5808.285419295379	4148.494855794077	3.021413130534454	2.7328507285115067
2016	TEN	18379.717610982843	18269.314063657068	18501.26278577736	0.0	0.0
2017	TEN	17687.27454894116	6069.9083783015285	6597.4763485591075	-66.7753898304462	-64.34039976108635
2018	TEN	18132.48562619872	18118.753628749215	18813.45669255206	198.5012705219642	185.16141170647666
2019	TEN	18118.524559088506	18064.151703945216	18370.295091800836	-0.30135585439697044	-2.355556493383081
2020	TEN	17427.250052932675	17440.703020531048	17375.86328129425	-3.4513034081639535	-5.413259860754381
2021	TEN	18259.087211197188	17084.39946716264	17822.738477498708	-2.042942609302912	2.5718157939556097
2016	MIDW	22783.070077217544	22566.054309174506	19281.491903040776	0.0	0.0
2017	MIDW	22374.921510233533	22404.803609360497	18776.489441237525	-0.7145719743679354	-2.6191047059154697

only showing top 20 rows