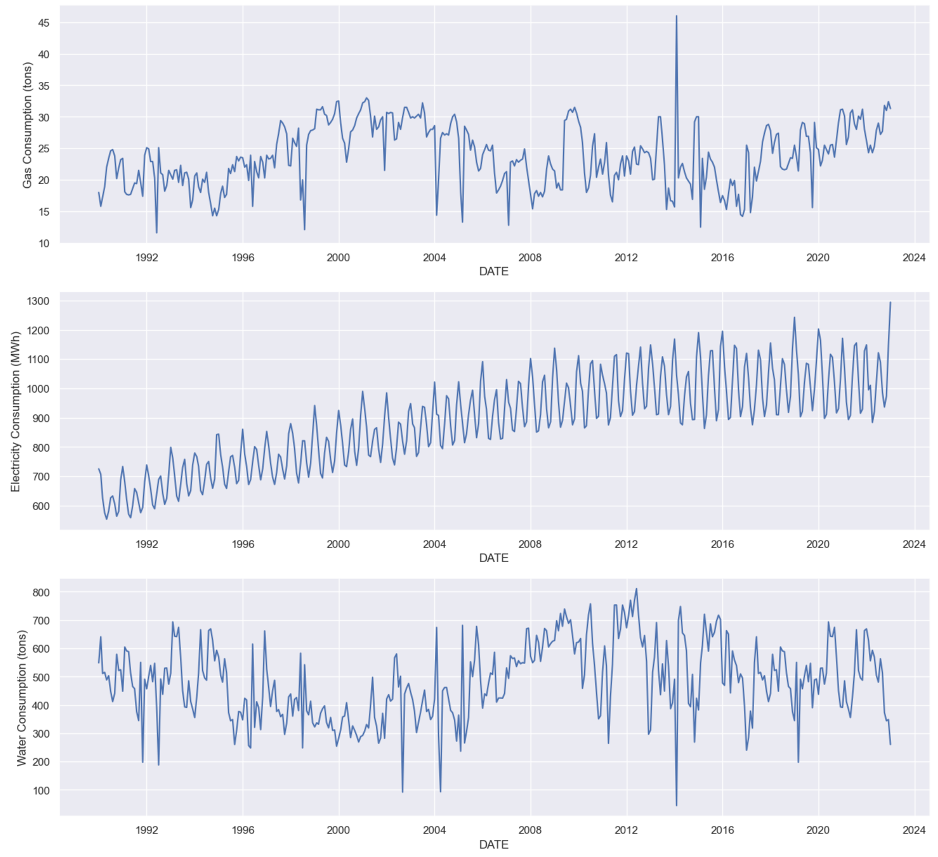
AIML CA2: Part A Time Series

Name: Glenn Wu

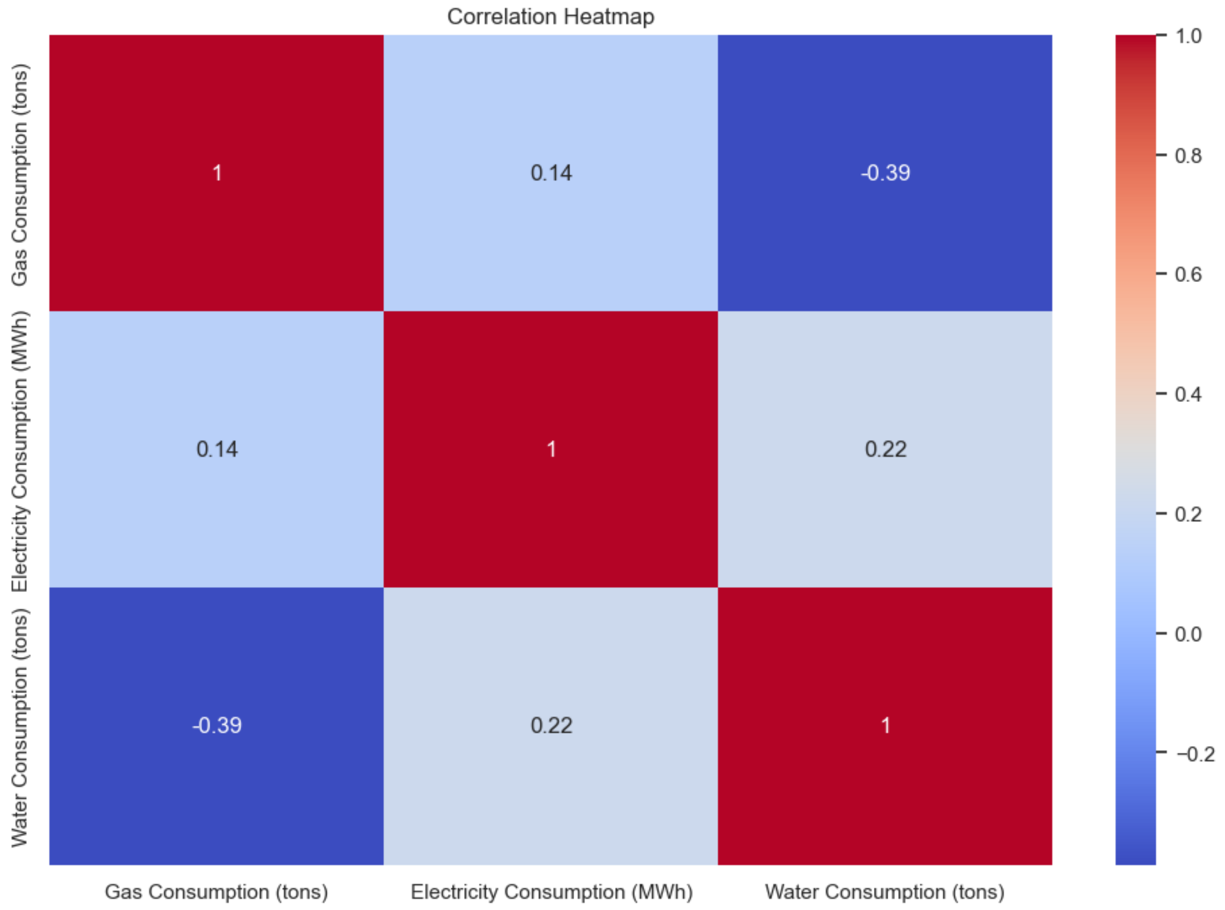
Admin No.: 2214395

Class: DAAA/FT/2A/01

Data Exploration

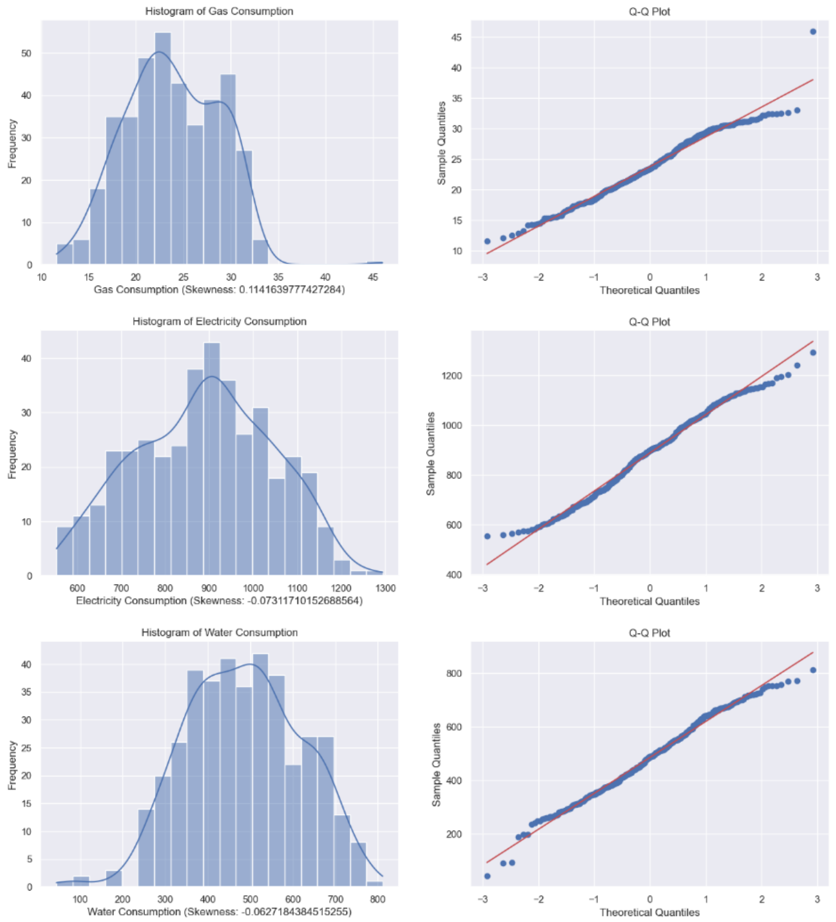


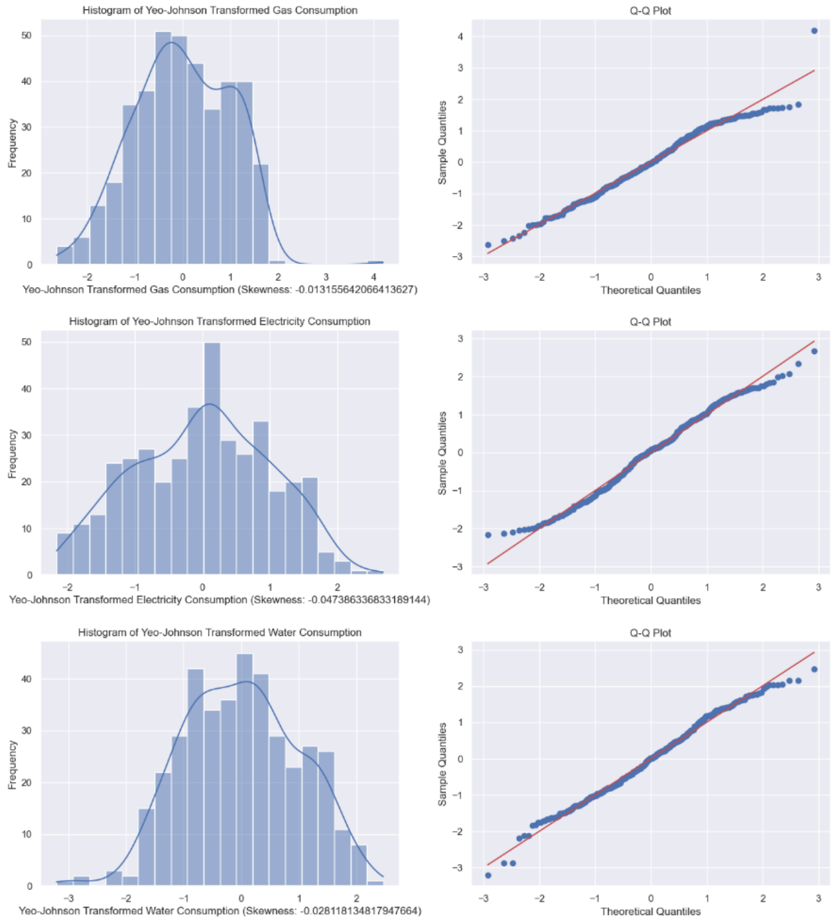
- Clear seasonal component for Electricity Consumption
- None of the time series look stationary



- Moderate negative linear relationship between Gas and Water consumption

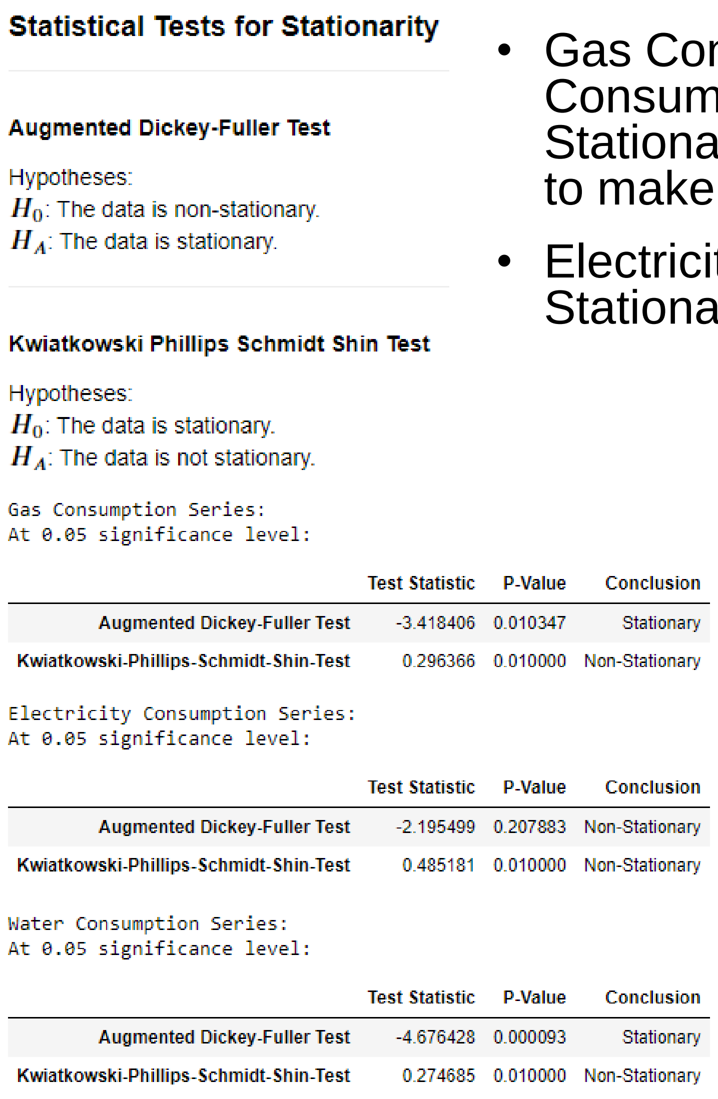
Data Exploration





- Reduce skewness of the time series using Yeo-Johnson Transformation. Data has very low skew to begin with.

Data Exploration: Stationarity and Differencing



Statistical Tests for Stationarity

Augmented Dickey-Fuller Test

Hypotheses:
H_0: The data is non-stationary.
H_A: The data is stationary.

Kwiatkowski Phillips Schmidt Shin Test

Hypotheses:
H_0: The data is stationary.
H_A: The data is not stationary.

```
Gas Consumption Series:
At 0.05 significance level:
```

	Test Statistic	P-Value	Conclusion
Augmented Dickey-Fuller Test	-3.418406	0.010347	Stationary
Kwiatkowski-Phillips-Schmidt-Shin-Test	0.296366	0.010000	Non-Stationary

```
Electricity Consumption Series:
At 0.05 significance level:
```

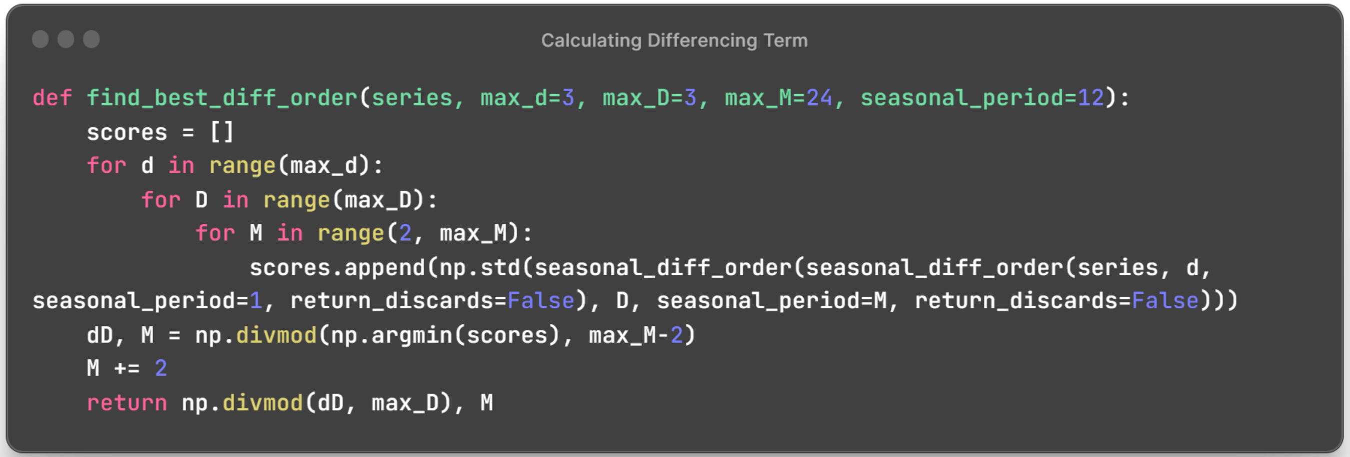
	Test Statistic	P-Value	Conclusion
Augmented Dickey-Fuller Test	-2.195499	0.207883	Non-Stationary
Kwiatkowski-Phillips-Schmidt-Shin-Test	0.485181	0.010000	Non-Stationary

```
Water Consumption Series:
At 0.05 significance level:
```

	Test Statistic	P-Value	Conclusion
Augmented Dickey-Fuller Test	-4.676428	0.000093	Stationary
Kwiatkowski-Phillips-Schmidt-Shin-Test	0.274685	0.010000	Non-Stationary

- Gas Consumption and Water Consumption: Difference Stationary, differencing is needed to make the data strictly stationary
- Electricity Consumption : Non-Stationary

I find the appropriate differencing term by iterating through a range of normal differencing and seasonal differencing orders and calculating the standard deviation for each. Then using the order with the lowest standard deviation.



Calculating Differencing Term

```python
def find_best_diff_order(series, max_d=3, max_D=3, max_M=24, seasonal_period=12):
    scores = []
    for d in range(max_d):
        for D in range(max_D):
            for M in range(2, max_M):
                scores.append(np.std(seasonal_diff_order(seasonal_diff_order(series, d,
seasonal_period=1, return_discards=False), D, seasonal_period=M, return_discards=False)))
    dD, M = np.divmod(np.argmin(scores), max_M-2)
    M += 2
    return np.divmod(dD, max_D), M
```

```
Gas Consumption Series Differencing Orders: ((1, 0), 2)
Electricity Consumption Series Differencing Orders: ((1, 1), 12)
Water Consumption Series Differencing Orders: ((1, 0), 2)
```

Data Exploration: Stationarity and Differencing

Statistical Tests for Stationarity

Augmented Dickey-Fuller Test

Hypotheses:
H_0: The data is non-stationary.
H_A: The data is stationary.

Kwiatkowski Phillips Schmidt Shin Test

Hypotheses:
H_0: The data is stationary.
H_A: The data is not stationary.

```
Gas Consumption Series:
At 0.05 significance level:
```

	Test Statistic	P-Value	Conclusion
Augmented Dickey-Fuller Test	-7.811089	7.066742e-12	Stationary
Kwiatkowski-Phillips-Schmidt-Shin-Test	0.089844	1.000000e-01	Stationary

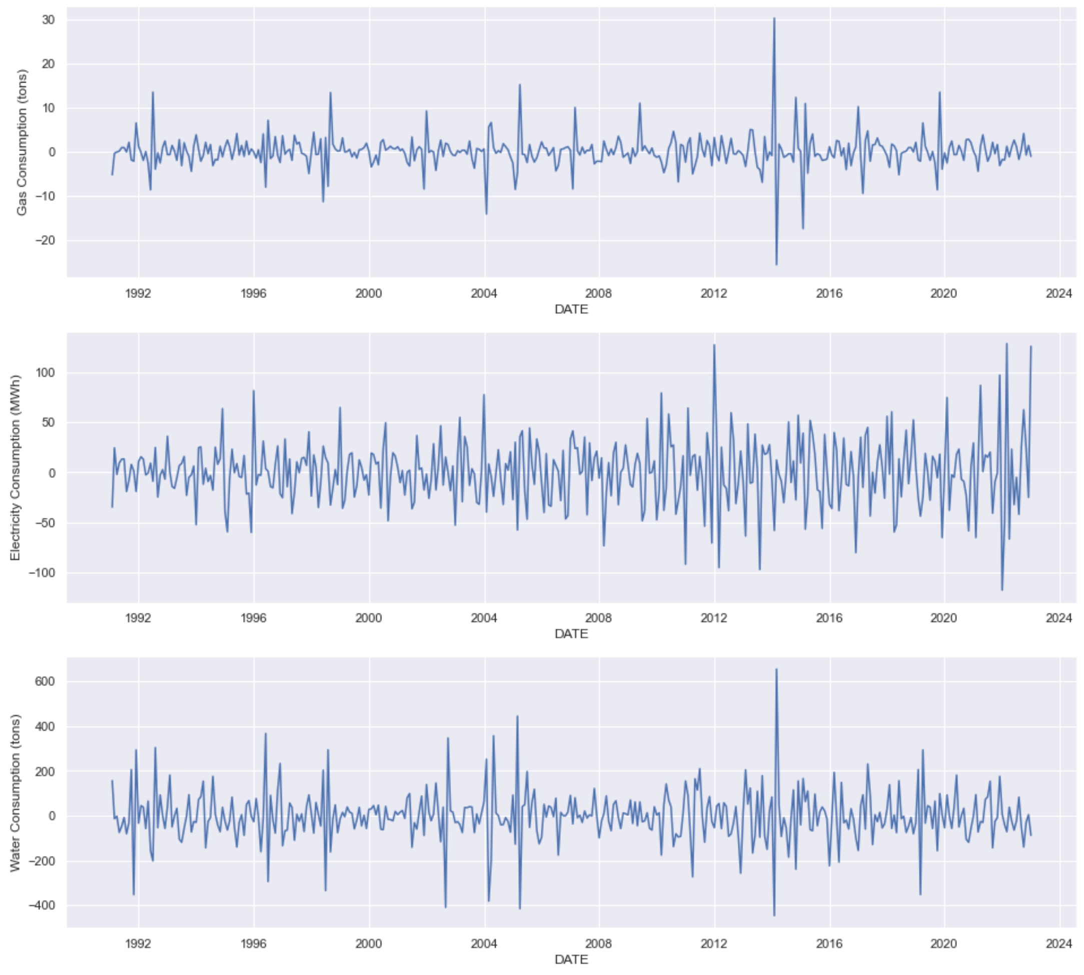
```
Electricity Consumption Series:
At 0.05 significance level:
```

	Test Statistic	P-Value	Conclusion
Augmented Dickey-Fuller Test	-8.019366	2.096457e-12	Stationary
Kwiatkowski-Phillips-Schmidt-Shin-Test	0.048818	1.000000e-01	Stationary

```
Water Consumption Series:
At 0.05 significance level:
```

	Test Statistic	P-Value	Conclusion
Augmented Dickey-Fuller Test	-9.062086	4.534434e-15	Stationary
Kwiatkowski-Phillips-Schmidt-Shin-Test	0.095551	1.000000e-01	Stationary

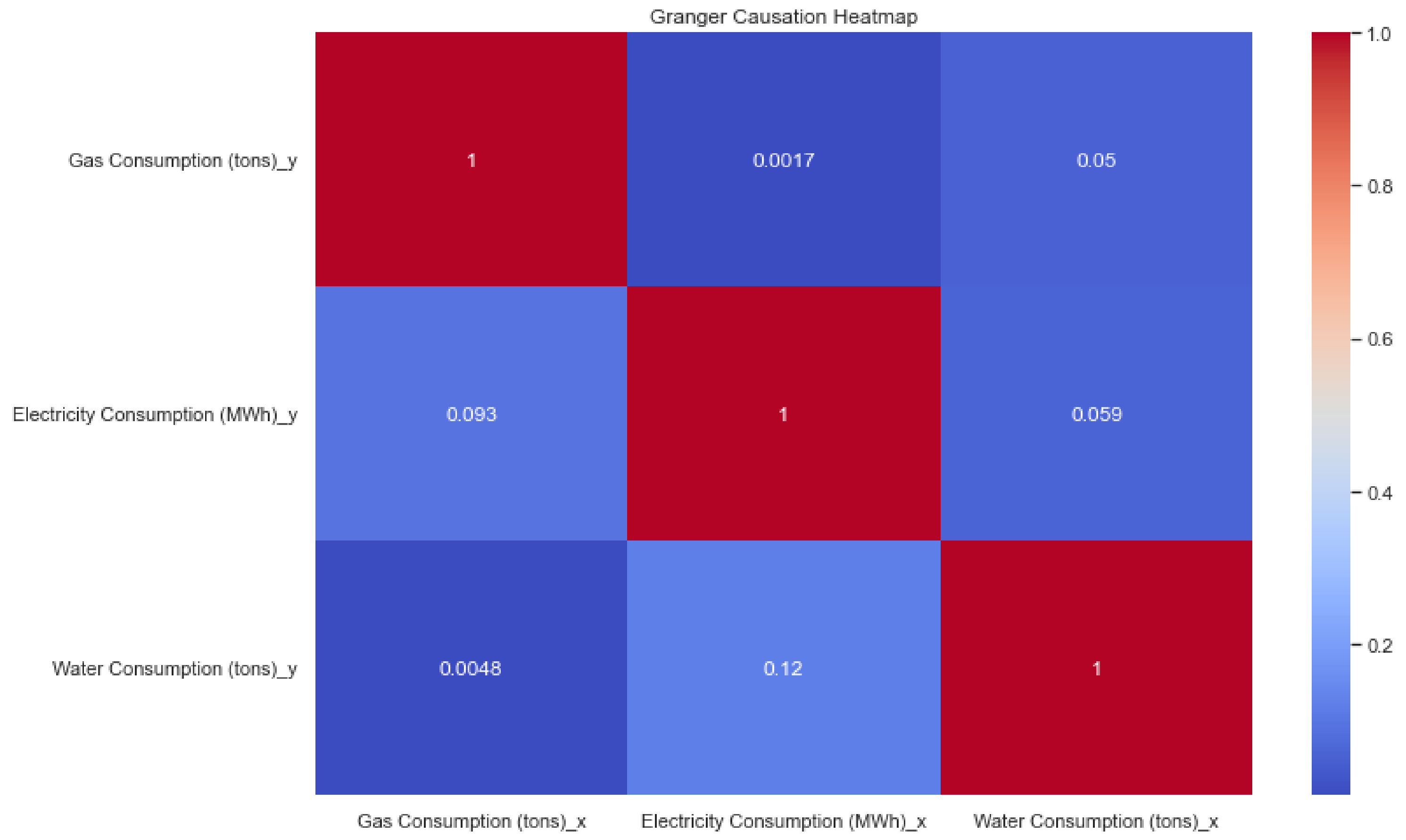
```
Gas Consumption Series Differencing Orders: ((1, 0), 2)
Electricity Consumption Series Differencing Orders: ((1, 1), 12)
Water Consumption Series Differencing Orders: ((1, 0), 2)
```



Data Exploration: Granger Causality Test

Granger Causality Test

Hypotheses:
H_0: The lagged values of variable X do not cause the values of variable Y.
H_A: The lagged values of variable X do Granger cause the values of variable Y.



- Based on a 0.05 significance level, we can conclude that <u>Gas Consumption causes Water Consumption</u> and <u>Electricity Consumption causes Gas Consumption.</u>

Model Selection

- Scoring Metrics:

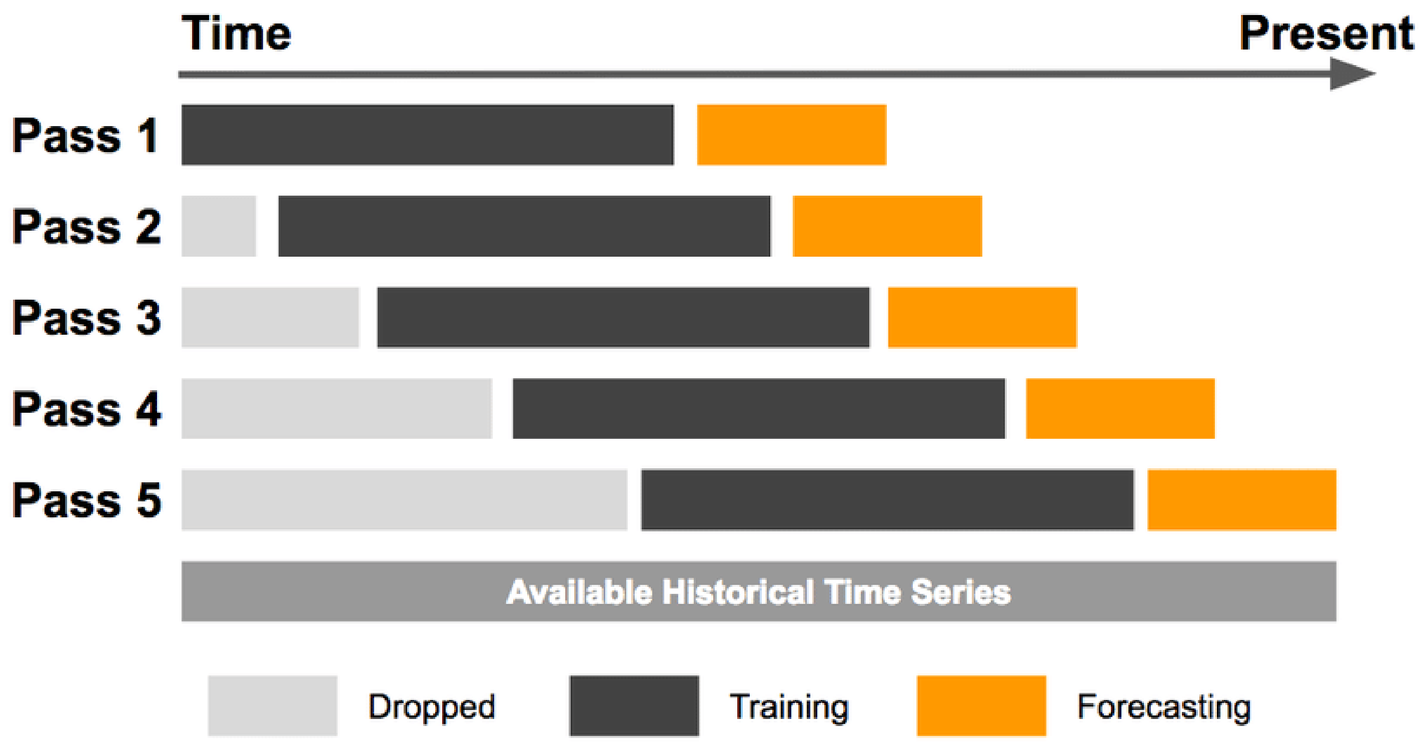
1. AIC (Akaike Information Criterion)
   - Formula: $\mathrm{AIC} = 2k - 2\ln(\hat{L})$
   - Legend:
     - k: Number of estimated parameters in the model.
     - $\hat{L}$: Maximum likelihood estimation of the model.

2. RMSE (Root Mean Square Error)
   - Formula: $\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$

3. MAE (Mean Absolute Error)
   - Formula: $\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$

Since the goal of the prediction task is to maximize accuracy for training and testing data, RMSE and MAE are my main focus. AIC does not capture prediction accuracy.

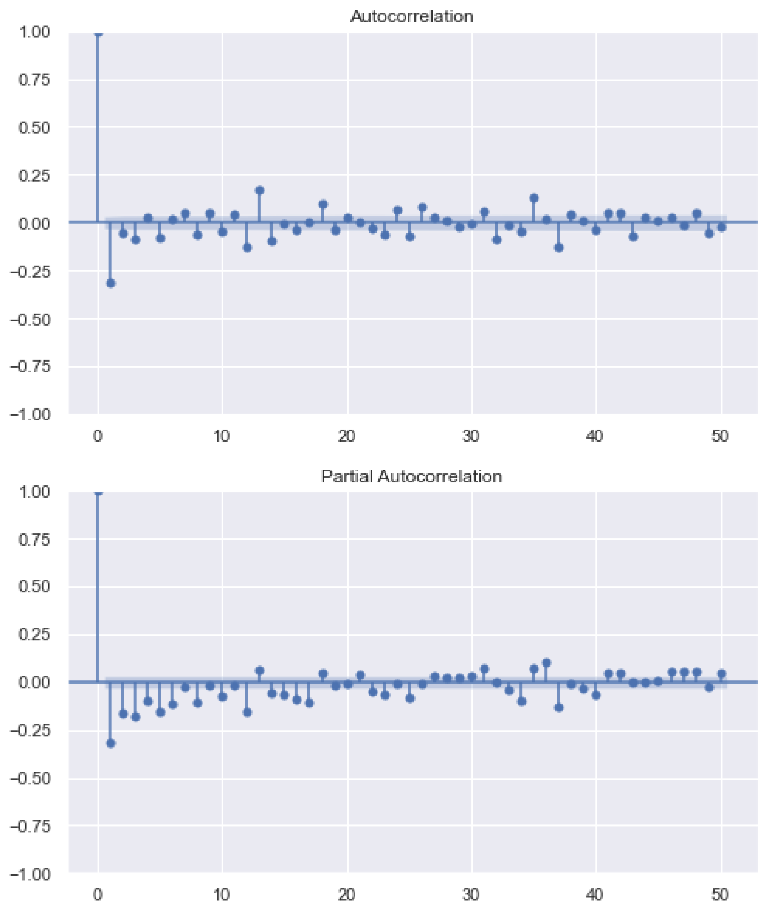
I found that some models and parameters produced a low AIC but scored worse in terms of RMSE and MAE compared to other models.

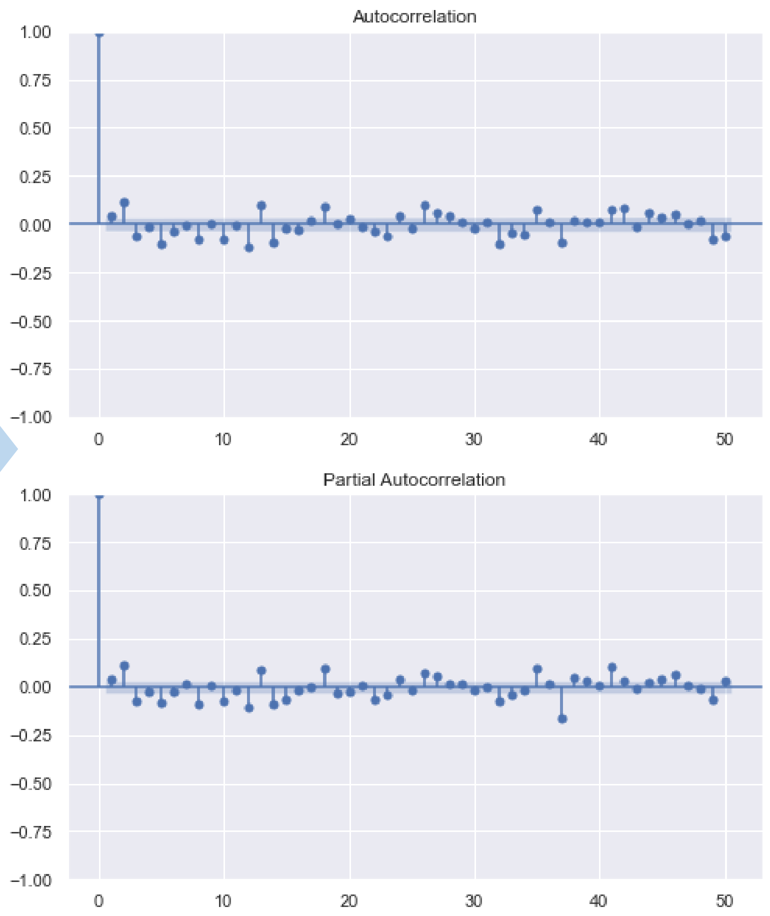


Rolling Walk-Forward Validation

Model Selection: Determining ARMA Terms

- Gas Consumption ARIMA(0, 1, 1)



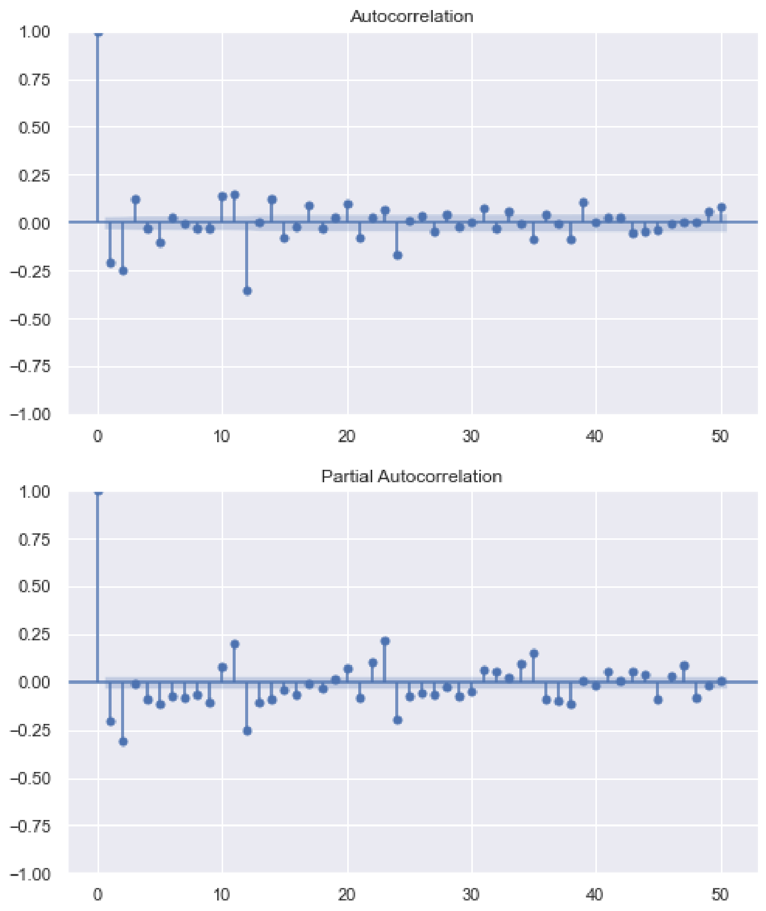


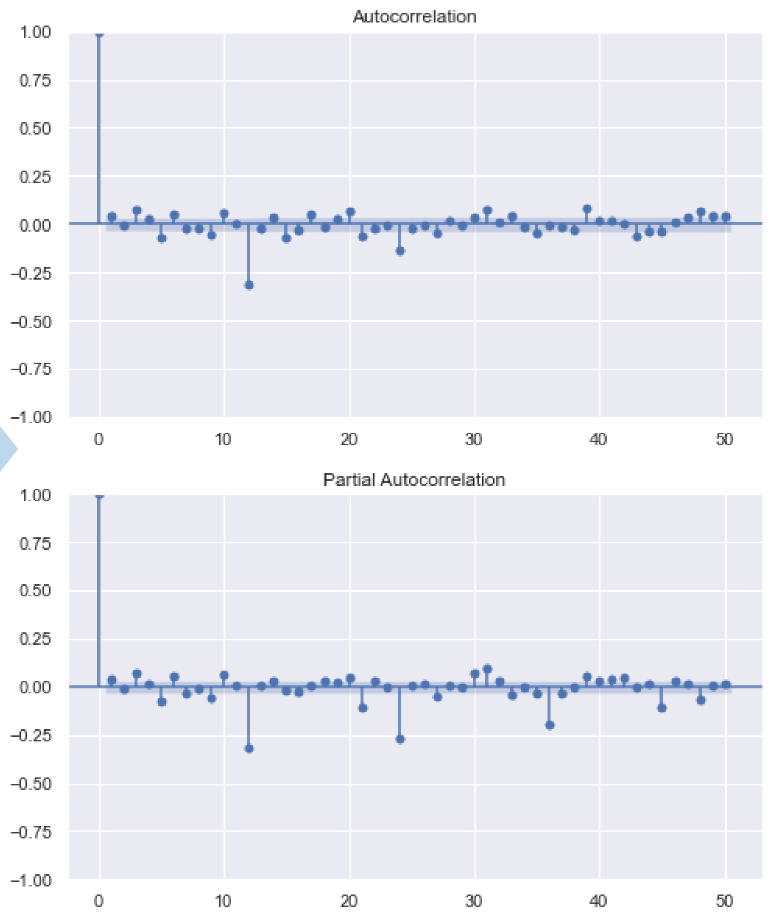
```
MAE (test):  2.4073503629944932
MAE (train):  2.485491324638523
RMSE (test):  3.118468218704852
RMSE (train):  3.7108166291410996
```

- Cutoff at lag-1 in ACF plot: add MA term

Model Selection: Determining ARMA Terms

- Electricity Consumption ARIMA(1, 1, 1)



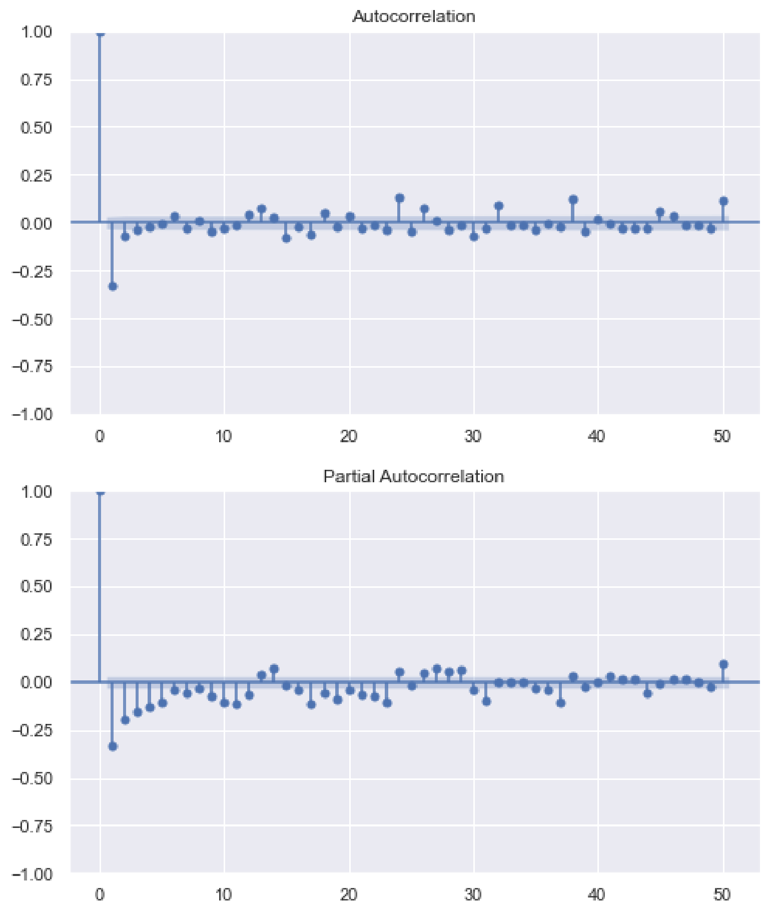


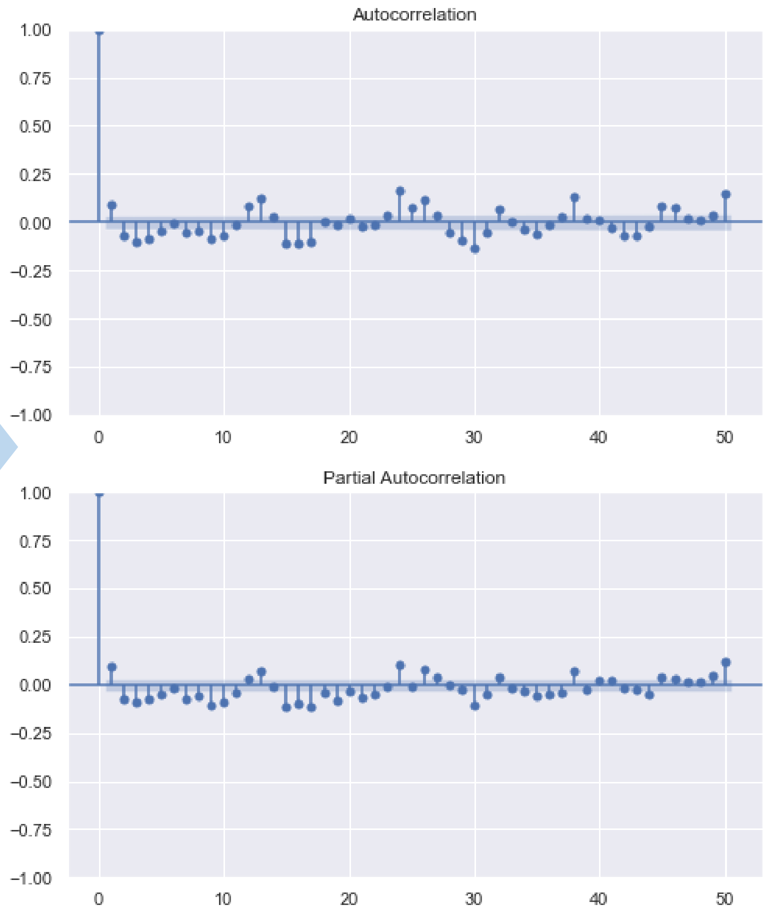
```
MAE (test):  33.4891156240216
MAE (train):  22.746346433399196
RMSE (test):  44.642696955118026
RMSE (train):  32.492074822577884
```

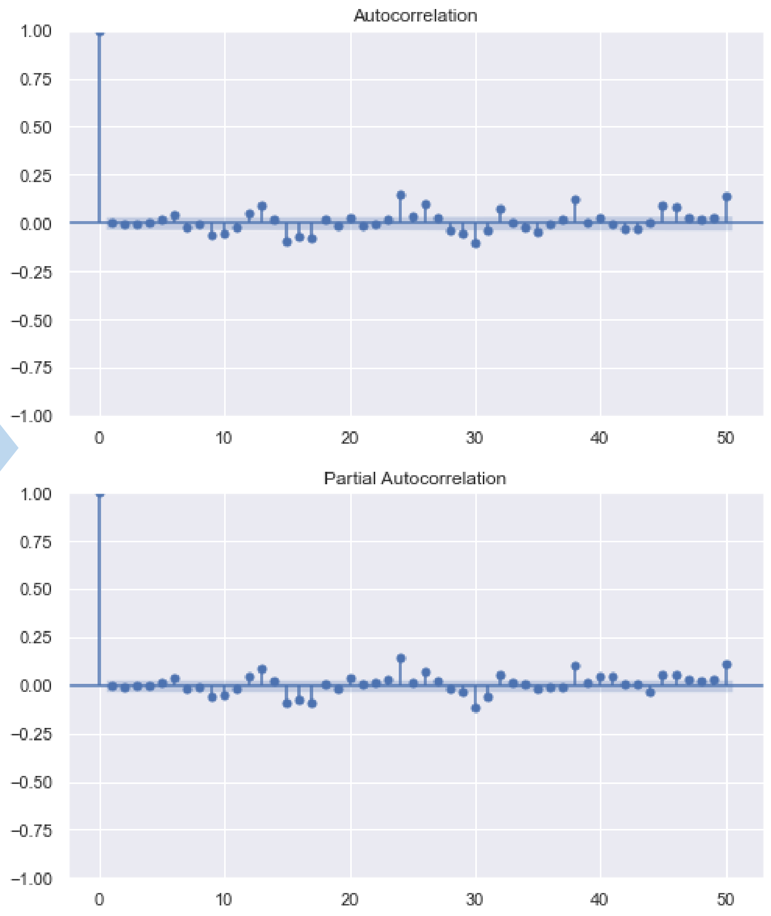
- Cutoff at lag-2 in both ACF and PACF plot: add AR and MA terms

Model Selection: Determining ARMA Terms

- Water Consumption ARIMA(1, 1, 2)





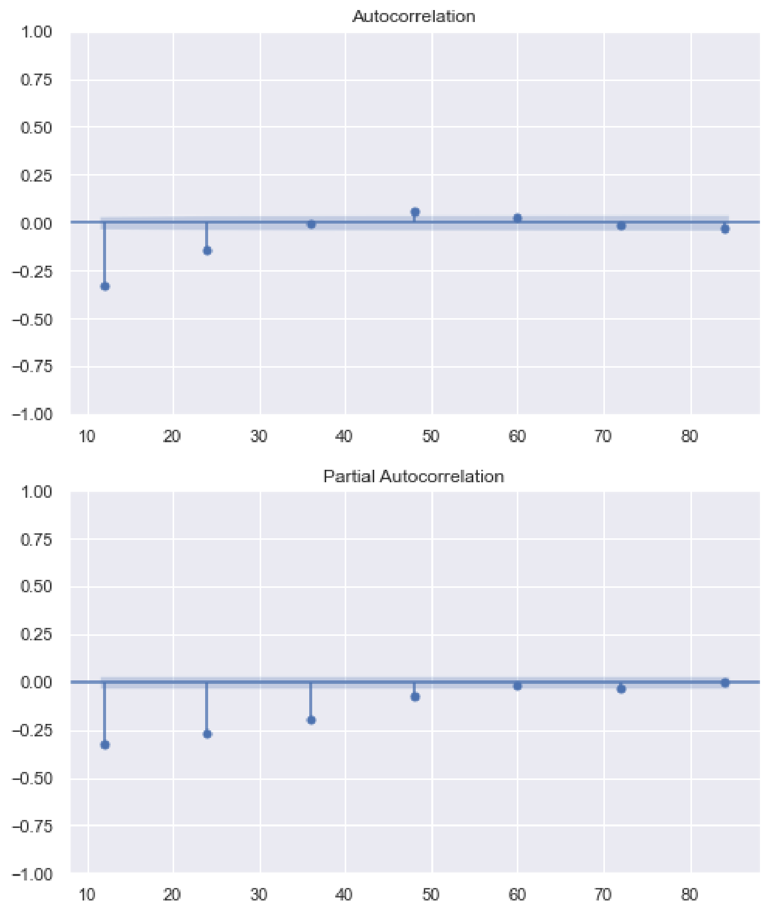


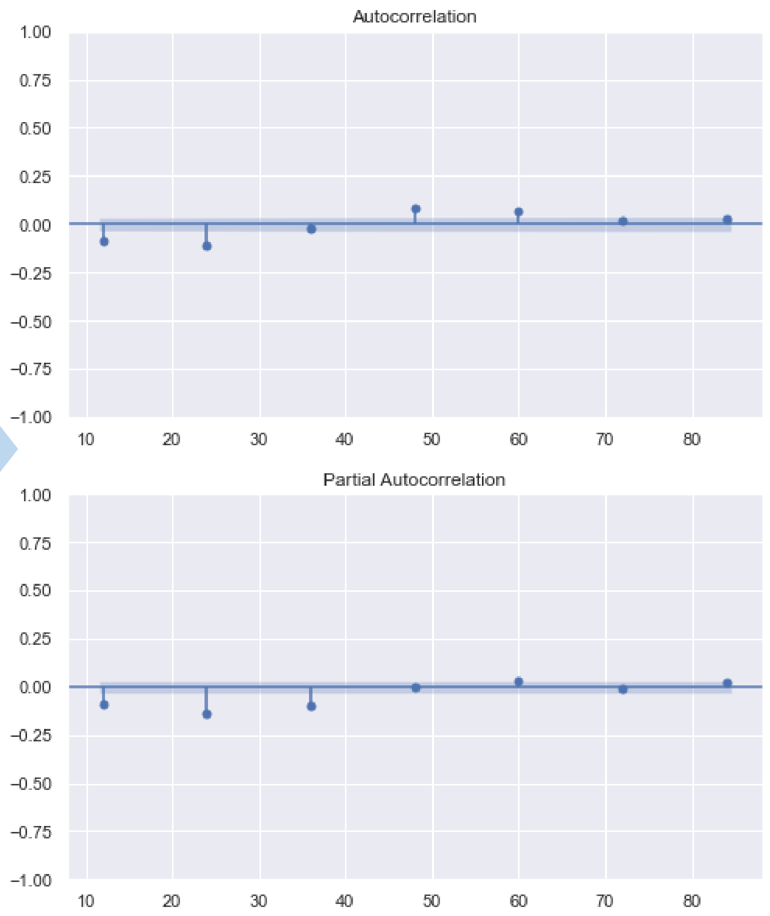
- Cutoff at lag-1 in ACF: add MA term
- Cutoff at lag-1 in both ACF and PACF plot: add AR and MA terms

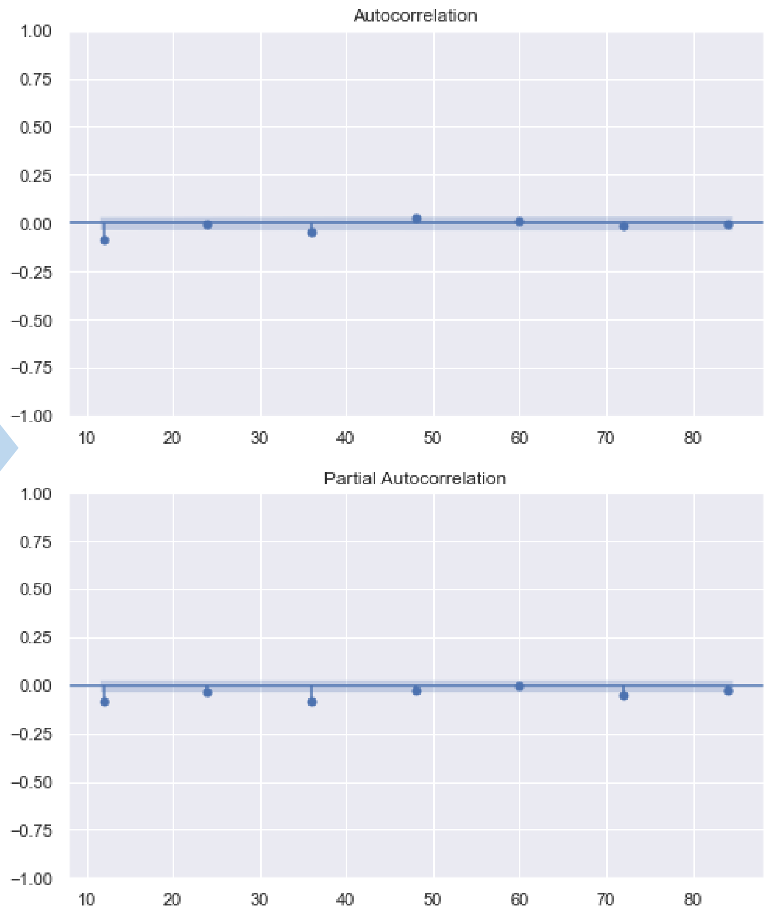
```
MAE (test):  87.29470570320926
MAE (train):  74.6029493326991
RMSE (test):  108.12251665263804
RMSE (train):  102.87341155545565
```

Model Selection: Determining SARMA Terms

- Electricity Consumption SARIMA(1, 1, 1) x (0, 1, 4, 12)







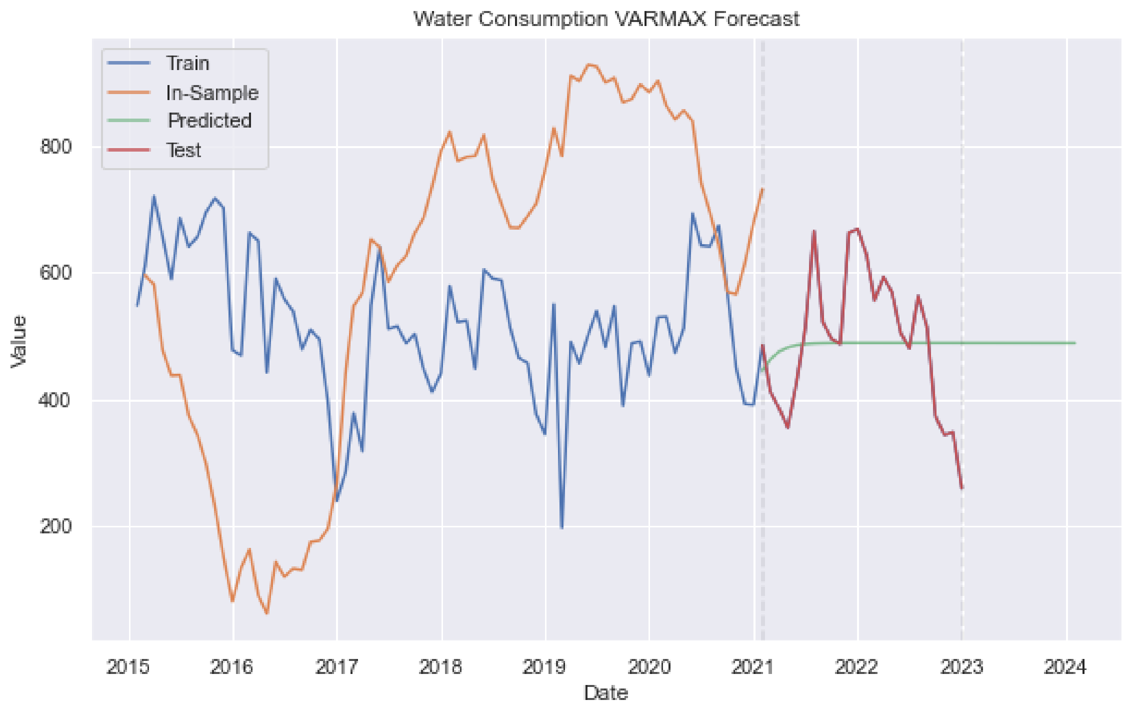
- Cutoff at lag-2 in ACF: add SMA term

- Cutoff at lag-2 in ACF plot: add more SMA terms

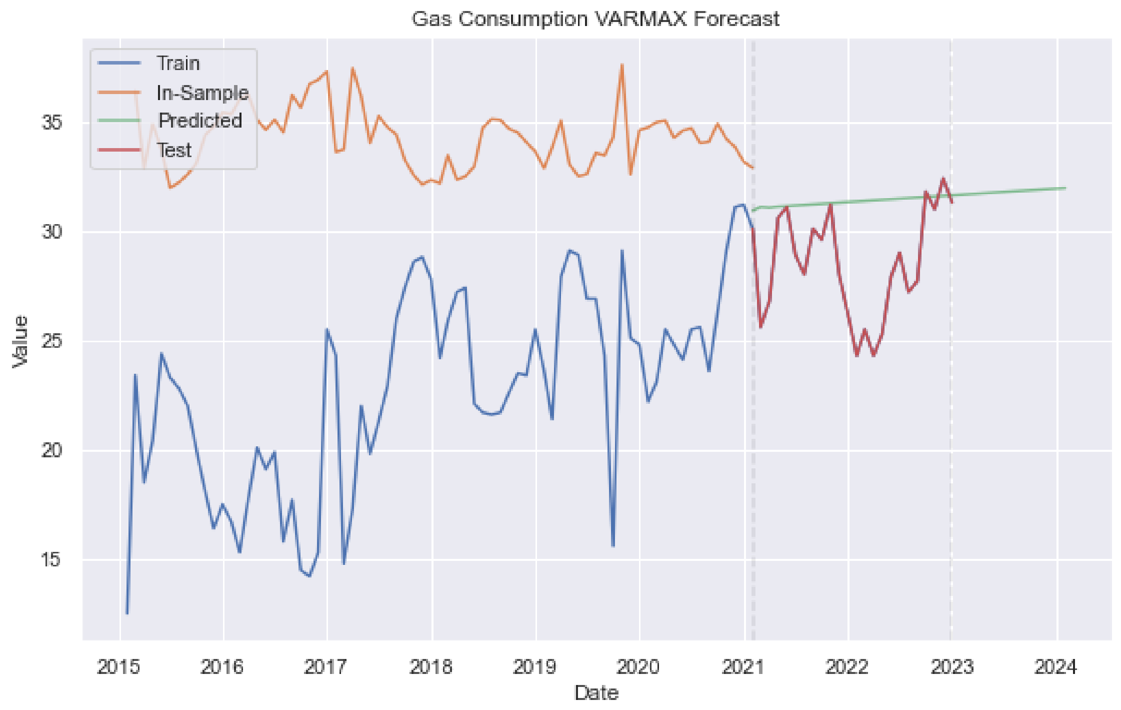
```
MAE (test):  27.227454608320993
MAE (train):  19.46576481135006
RMSE (test):  34.91644318244092
RMSE (train):  28.718048345060648
```

Model Selection: VARMA

- Water & Gas Consumption VARMA(2, 1)
- Gas & Electricity Consumption VARMA(1, 2)

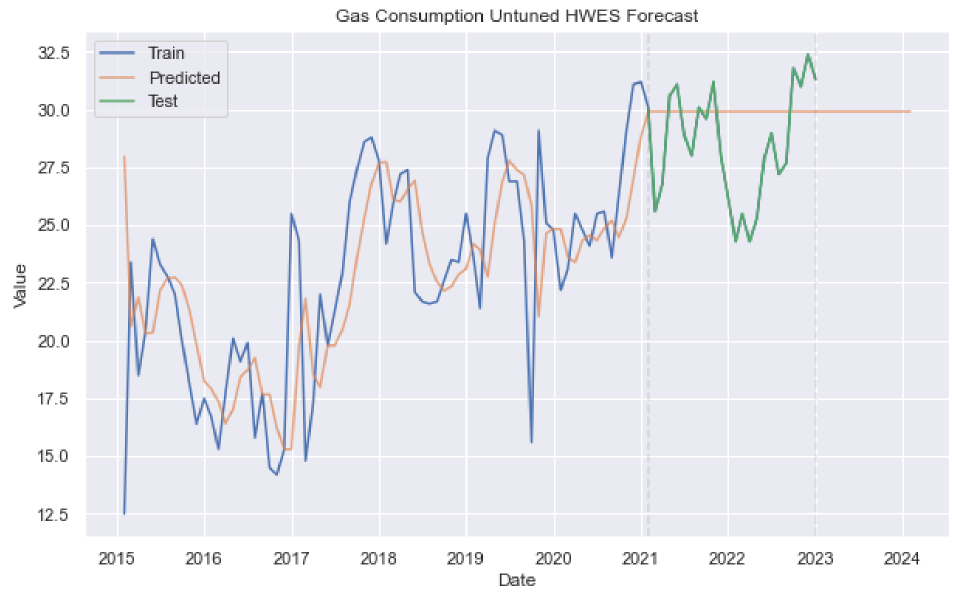


```
MAE (test):  60.9644393164263
MAE (train):  782.5407044448101
RMSE (test):  76.32959123852366
RMSE (train):  1006.8682814470981
```

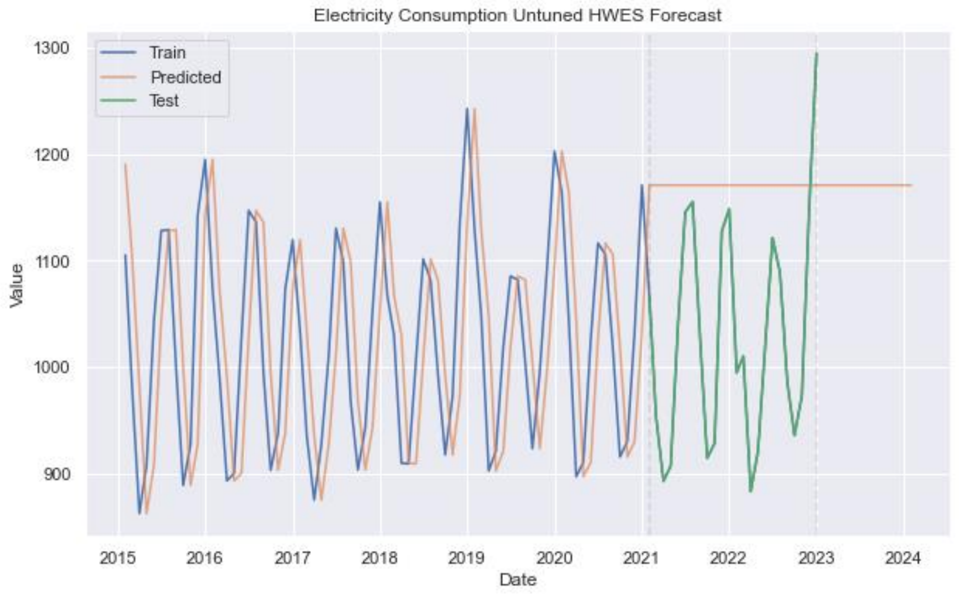


```
MAE (test):  2.649907593782821
MAE (train):  14.978199307181153
RMSE (test):  3.2873323741756915
RMSE (train):  18.214567647123523
```

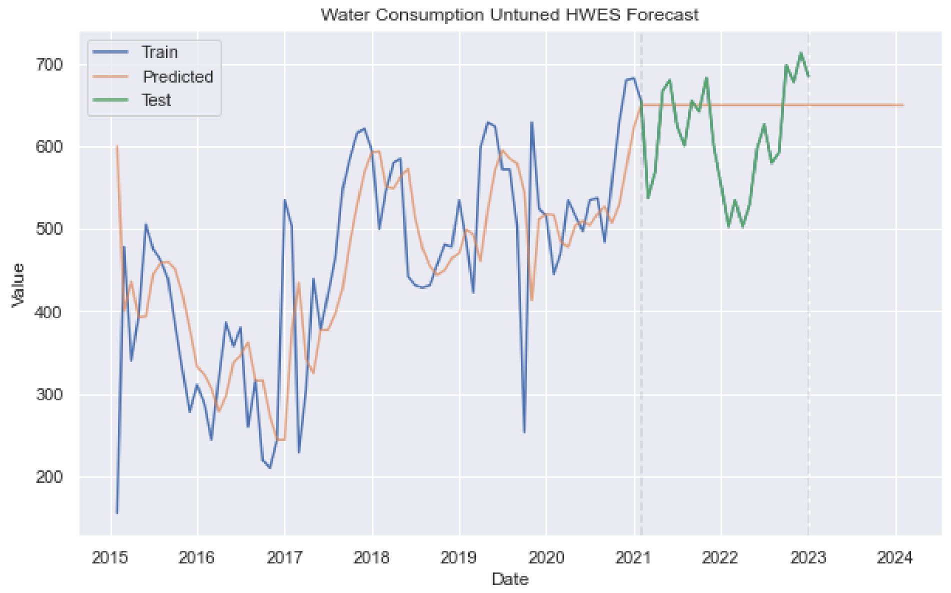
Model Selection: Holt-Winter's Exponential Smoothing



```
MAE (test): 2.283070889330169
RMSE (test): 2.799318605517012
```



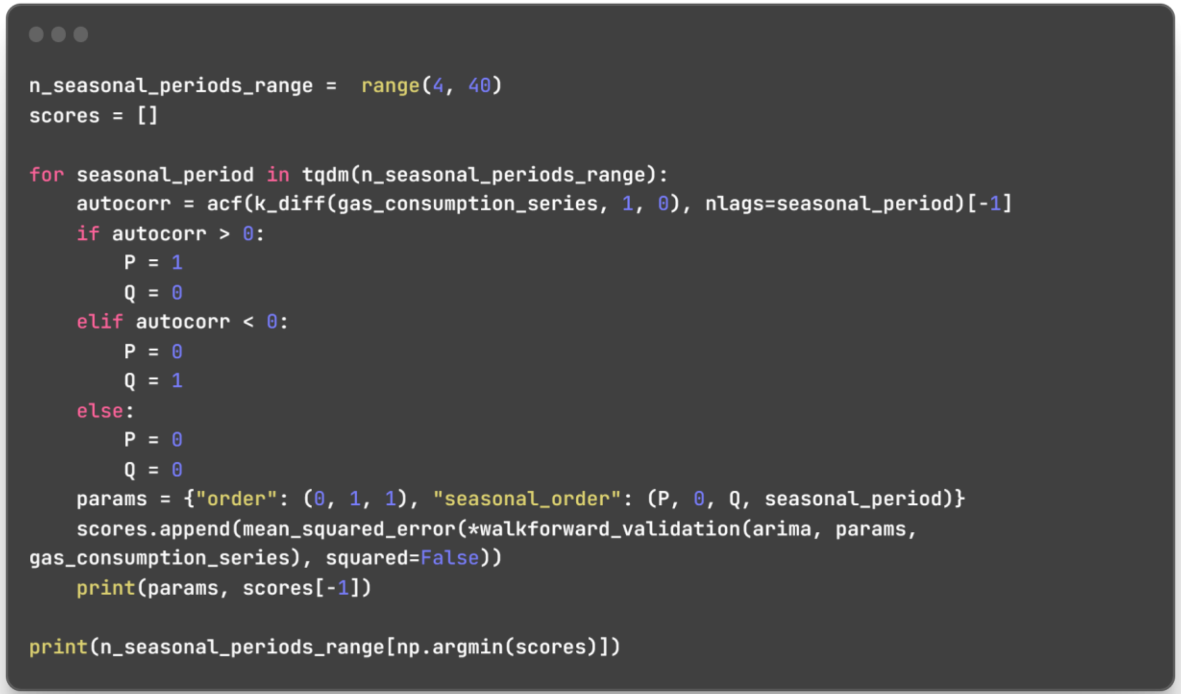
```
MAE (test): 152.38749810969824
RMSE (test): 176.3203566348815
```



```
MAE (test): 59.05157788424297
RMSE (test): 72.7657631048426
```

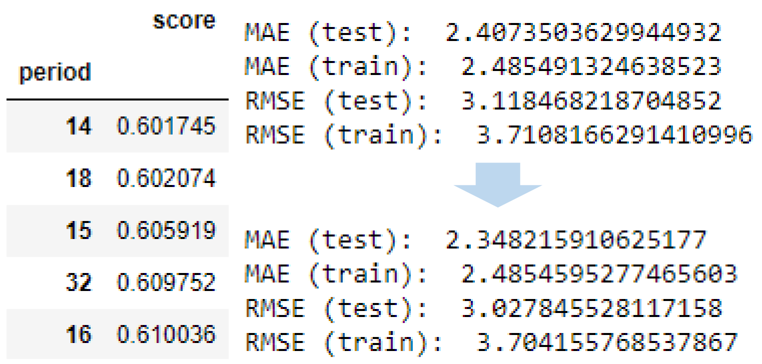
Model Improvement: Seasonality

- For Gas and Water which do not have an obvious seasonal period, I will iterate through a range of possible seasonal periods to determine which seasonal period produces the optimal forecast accuracy based on RMSE.
- For the SAR or SMA terms, I will follow this rule:
    - If the autocorrelation of the appropriately differenced series is positive at lag s, where s is the number of periods in a season, then consider adding an SAR term to the model. If the autocorrelation of the differenced series is negative at lag s, consider adding an SMA term to the model. From this article.



```python
n_seasonal_periods_range =  range(4, 40)
scores = []

for seasonal_period in tqdm(n_seasonal_periods_range):
    autocorr = acf(k_diff(gas_consumption_series, 1, 0), nlags=seasonal_period)[-1]
    if autocorr > 0:
        P = 1
        Q = 0
    elif autocorr < 0:
        P = 0
        Q = 1
    else:
        P = 0
        Q = 0
    params = {"order": (0, 1, 1), "seasonal_order": (P, 0, Q, seasonal_period)}
    scores.append(mean_squared_error(*walkforward_validation(arima, params,
gas_consumption_series), squared=False))
    print(params, scores[-1])

print(n_seasonal_periods_range[np.argmin(scores)])
```

Gas SARIMA(0, 1, 1) x (0, 0, 1, 14)

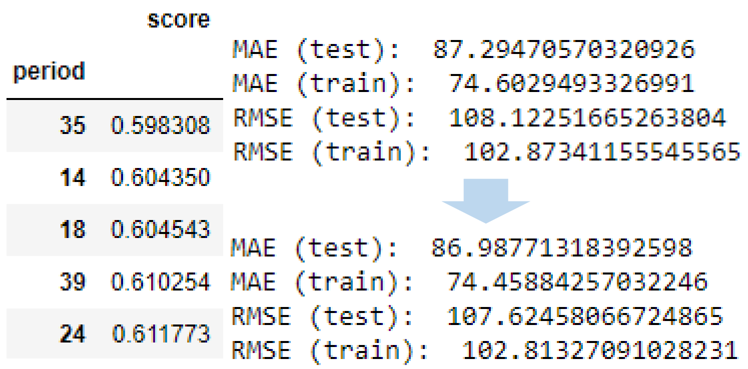


period	score
14	0.601745
18	0.602074
15	0.605919
32	0.609752
16	0.610036

```
MAE (test):  2.4073503629944932
MAE (train):  2.485491324638523
RMSE (test):  3.118468218704852
RMSE (train):  3.7108166291410996
```

```
MAE (test):  2.348215910625177
MAE (train):  2.485495277465603
RMSE (test):  3.027845528117158
RMSE (train):  3.704155768537867
```

Water SARIMA(1, 1, 2) x (0, 0, 1, 35)



period	score
35	0.598308
14	0.604350
18	0.604543
39	0.610254
24	0.611773

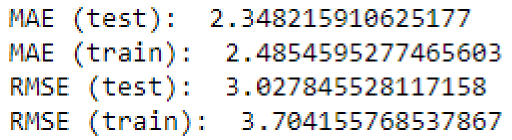
```
MAE (test):  87.29470570320926
MAE (train):  74.6029493326991
RMSE (test):  108.12251665263804
RMSE (train):  102.87341155545565
```

```
MAE (test):  86.98771318392598
MAE (train):  74.45884257032246
RMSE (test):  107.62458066724865
RMSE (train):  102.81327091028231
```

Model Improvement: Auto ARIMA

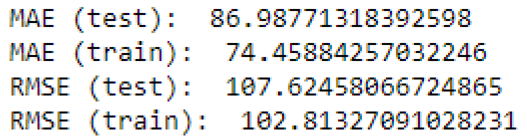
- Auto ARIMA is a tuning function that finds the optimal combination of p, d, q, P, D, and Q based on the AIC score
- It can be useful to compare its results to our results but since it uses AIC, I still have to compare using RMSE and MAE

Original: Gas SARIMA(0, 1, 1) x (0, 0, 1, 14)



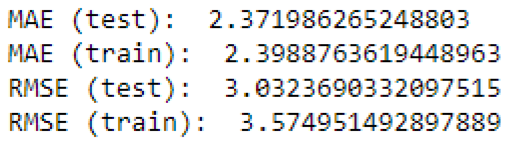
```
MAE (test):  2.348215910625177
MAE (train):  2.4854595277465603
RMSE (test):  3.027845528117158
RMSE (train):  3.704155768537867
```

Original: Water SARIMA(1, 1, 2) x (0, 0, 1, 35)



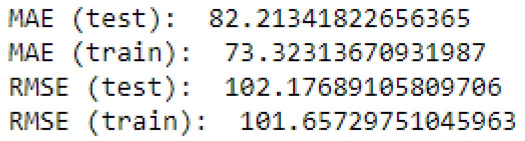
```
MAE (test):  86.98771318392598
MAE (train):  74.45884257032246
RMSE (test):  107.62458066724865
RMSE (train):  102.81327091028231
```

Auto ARIMA: Gas SARIMA(1, 1, 1) x (1, 0, 0, 14)



```
MAE (test):  2.371986265248803
MAE (train):  2.3988763619448963
RMSE (test):  3.0323690332097515
RMSE (train):  3.574951492897889
```

Auto ARIMA: Water SARIMA(1, 1, 2) x (2, 0, 0, 12)



```
MAE (test):  82.21341822656365
MAE (train):  73.32313670931987
RMSE (test):  102.17689105809706
RMSE (train):  101.65729751045963
```

- Auto ARIMA gave the same order as before for Electricity consumption.

Model Improvement: Hyperparameter Tuning

- Artificial Bee Colony (ABC) Algorithm is a nature-inspired metaheuristic optimization algorithm that mimics the foraging behavior of honeybees.
- O(n) time complexity compared to other methods like Genetic Algorithm with O(n^2) time complexity.

Original: Gas SARIMA(0, 1, 1) x (0, 0, 1, 14)

```
MAE (test):  2.348215910625177
MAE (train):  2.4854595277465603
RMSE (test):  3.027845528117158
RMSE (train):  3.704155768537867
```

Auto ARIMA: Water SARIMA(1, 1, 2) x (2, 0, 0, 12)

```
MAE (test):  82.21341822656365
MAE (train):  73.32313670931987
RMSE (test):  102.17689105809706
RMSE (train):  101.65729751045963
```

Original: Electricity SARIMA(1, 1, 1) x (0, 1, 4, 12)

```
MAE (test):  27.22745460832099З
MAE (train):  19.46576481135006
RMSE (test):  34.91644318244092
RMSE (train):  28.718048345060648
```

ABC: Gas SARIMA(0, 1, 2) x (0, 0, 1, 14)

```
MAE (test):  2.3576847707013817
MAE (train):  2.4699365847489494
RMSE (test):  2.9968104251050356
RMSE (train):  3.6343260417374985
```

ABC: Water SARIMA(1, 1, 2) x (2, 0, 0, 12)

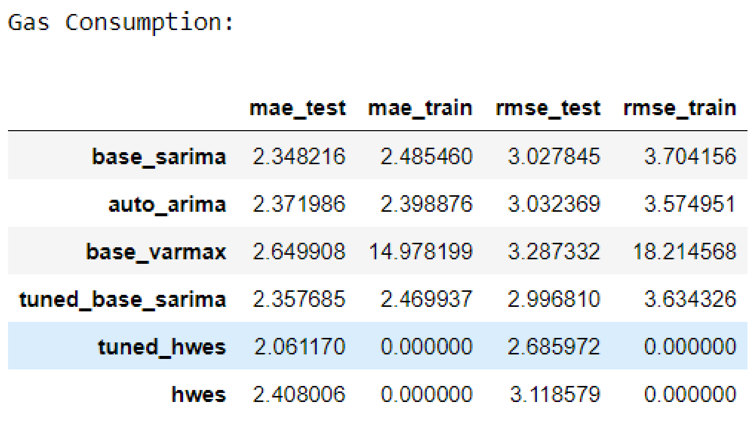
```
MAE (test):  80.06522230790436
MAE (train):  73.61789267938417
RMSE (test):  99.72139237920337
RMSE (train):  101.60219886599555
```

ABC: Electricity SARIMA(1, 1, 1) x (0, 1, 4, 12)

```
MAE (test):  26.11821889315З255
MAE (train):  20.57441427838683
RMSE (test):  34.02754780177266
RMSE (train):  30.522335748783238
```

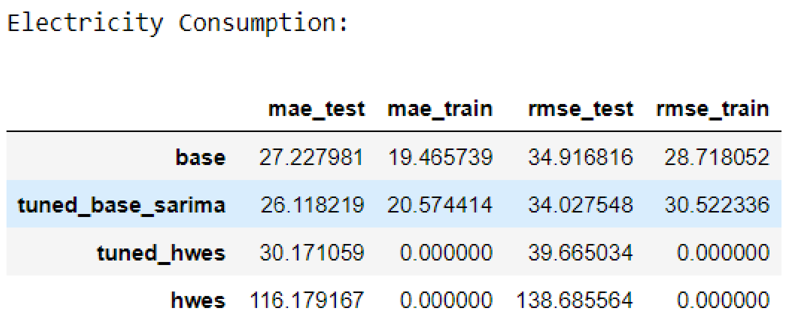
```
Gas Consumption Parameters:
 {'trend': 'c', 'measurement_error': True, 'simple_differencing': False, 'enforce_stationarity': False, 'enforce_invertibilit
y': True, 'trend_offset': 4, 'order': (0, 1, 2), 'seasonal_order': (0, 0, 1, 14)}

Electricity Consumption Parameters:
 {'trend': 'c', 'measurement_error': True, 'simple_differencing': False, 'enforce_stationarity': False, 'enforce_invertibilit
y': False, 'trend_offset': 2, 'order': (1, 1, 1), 'seasonal_order': (0, 1, 4, 12)}

Water Consumption Parameters:
 {'trend': 'n', 'measurement_error': True, 'simple_differencing': False, 'enforce_stationarity': True, 'enforce_invertibility':
True, 'trend_offset': 4, 'order': (1, 1, 2), 'seasonal_order': (2, 0, 0, 12)}
```

Conclusion



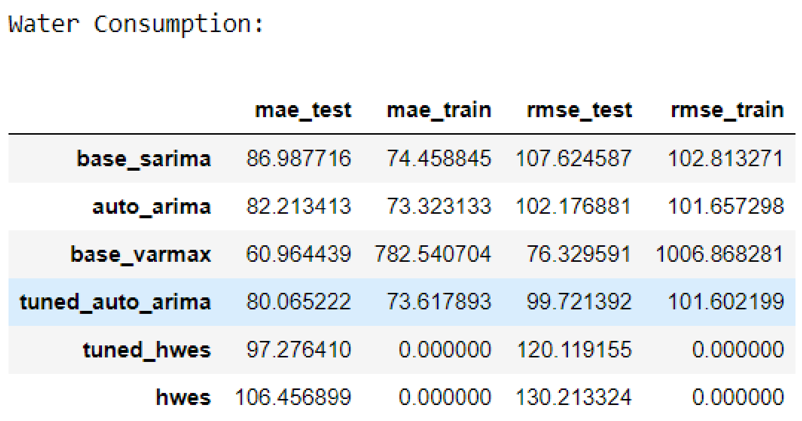
```
Gas Consumption:
```

	mae_test	mae_train	rmse_test	rmse_train
base_sarima	2.348216	2.485460	3.027845	3.704156
auto_arima	2.371986	2.398876	3.032369	3.574951
base_varmax	2.649908	14.978199	3.287332	18.214568
tuned_base_sarima	2.357685	2.469937	2.996810	3.634326
tuned_hwes	2.061170	0.000000	2.685972	0.000000
hwes	2.408006	0.000000	3.118579	0.000000



```
Electricity Consumption:
```

	mae_test	mae_train	rmse_test	rmse_train
base	27.227981	19.465739	34.916816	28.718052
tuned_base_sarima	26.118219	20.574414	34.027548	30.522336
tuned_hwes	30.171059	0.000000	39.665034	0.000000
hwes	116.179167	0.000000	138.685564	0.000000

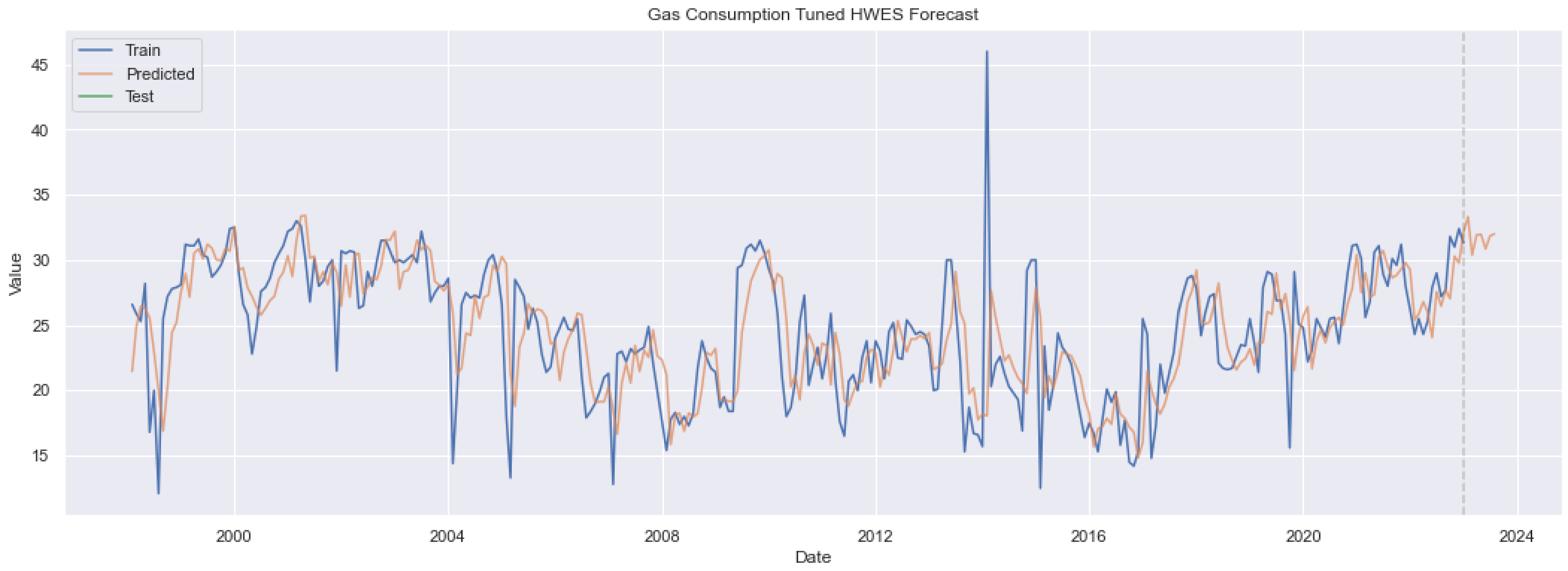


```
Water Consumption:
```

	mae_test	mae_train	rmse_test	rmse_train
base_sarima	86.987716	74.458845	107.624587	102.813271
auto_arima	82.213413	73.323133	102.176881	101.657298
base_varmax	60.964439	782.540704	76.329591	1006.868281
tuned_auto_arima	80.065222	73.617893	99.721392	101.602199
tuned_hwes	97.276410	0.000000	120.119155	0.000000
hwes	106.456899	0.000000	130.213324	0.000000

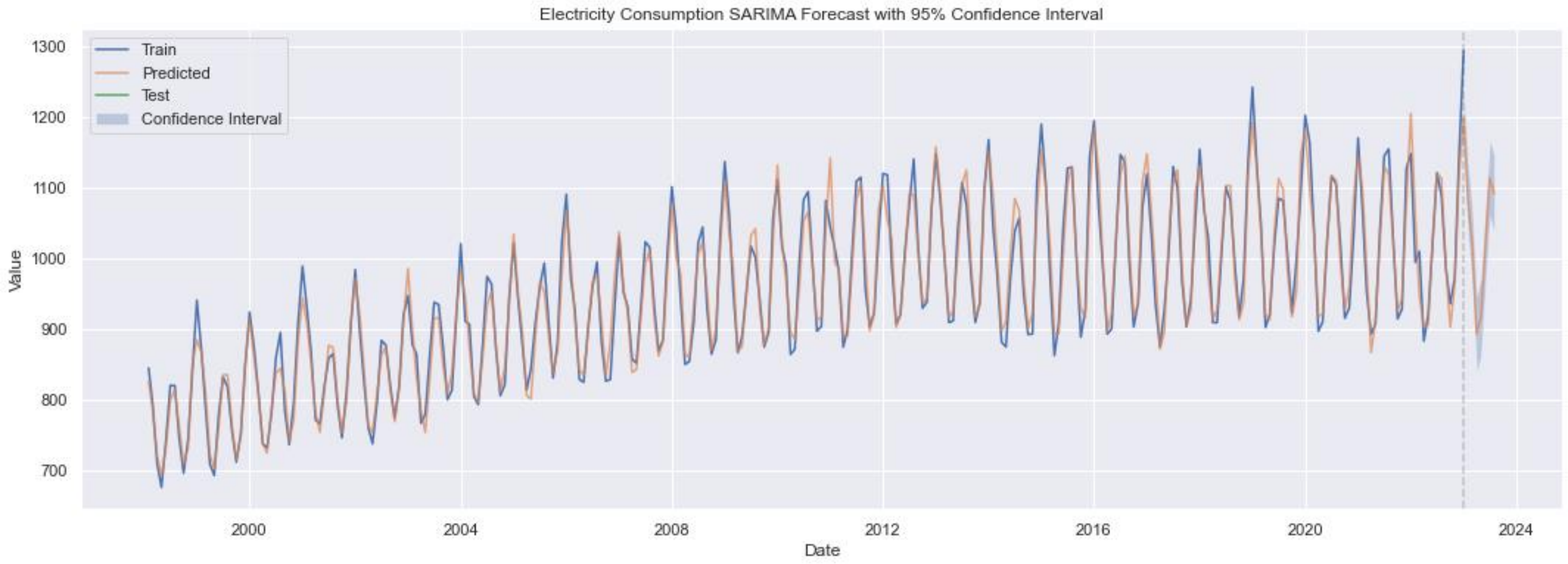
- Tuned HWES performed the best for Gas Consumption
- Tuned SARIMA performed the best for Electricity Consumption
- Even though VARMAX performed the best in terms of RMSE and MAE for Water Consumption, the very bad training score as well as irregular plot makes me reject it in favour of SARIMA

Conclusion: Gas Consumption



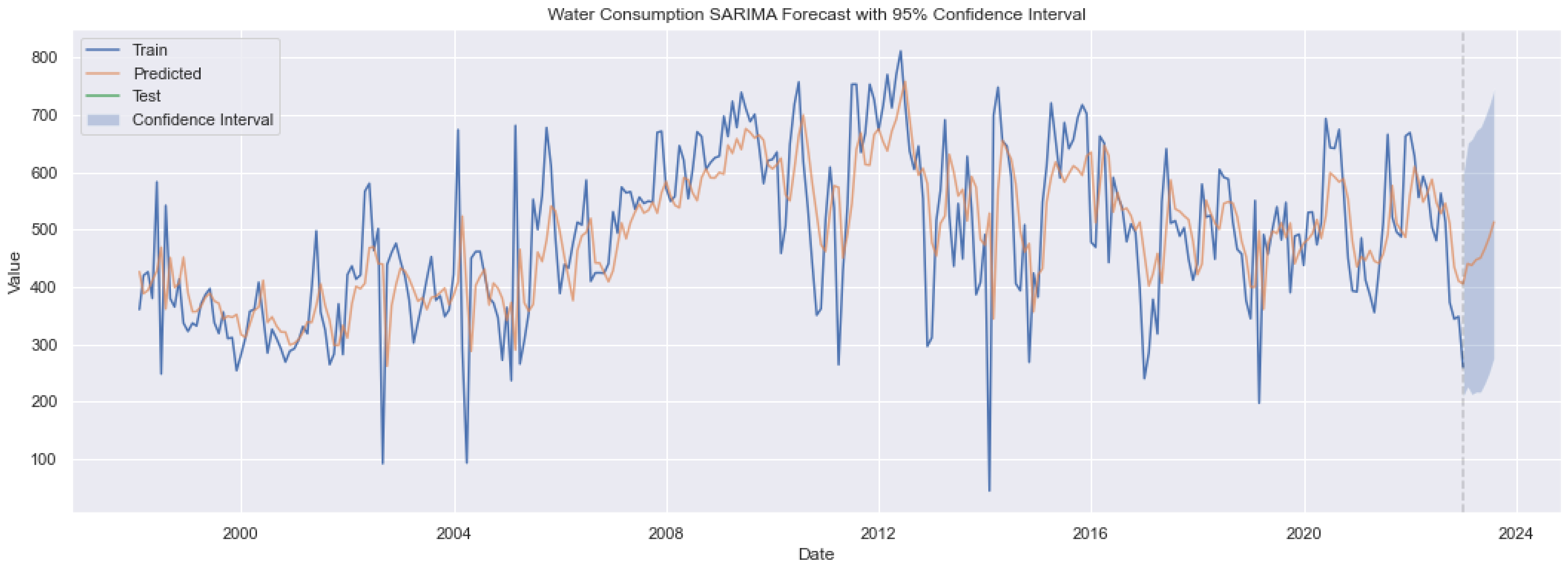
```
Gas Consumption Parameters:
 {'trend': 'add', 'damped_trend': True, 'seasonal': 'add', 'seasonal_periods': 36, 'initialization_method': 'estimated', 'use_b
oxcox': False}
```

Conclusion: Electricity Consumption



```
Electricity Consumption Parameters:
 {'trend': 'c', 'measurement_error': True, 'simple_differencing': False, 'enforce_stationarity': False, 'enforce_invertibilit
y': False, 'trend_offset': 2, 'order': (1, 1, 1), 'seasonal_order': (0, 1, 4, 12)}
```

Conclusion: Water Consumption



```
Water Consumption Parameters:
 {'trend': 'n', 'measurement_error': True, 'simple_differencing': False, 'enforce_stationarity': True, 'enforce_invertibility':
True, 'trend_offset': 4, 'order': (1, 1, 2), 'seasonal_order': (2, 0, 0, 12)}
```

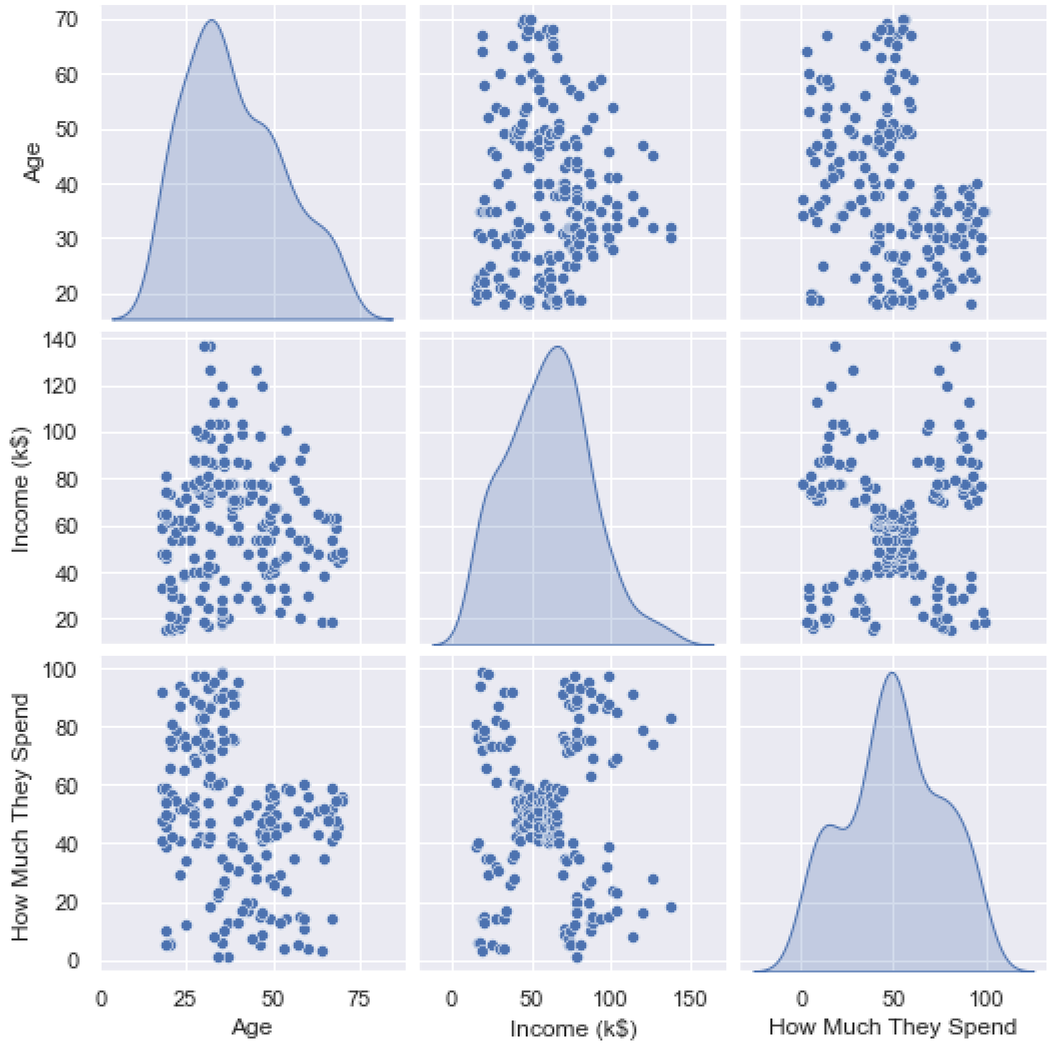
AIML CA2: Part B Clustering

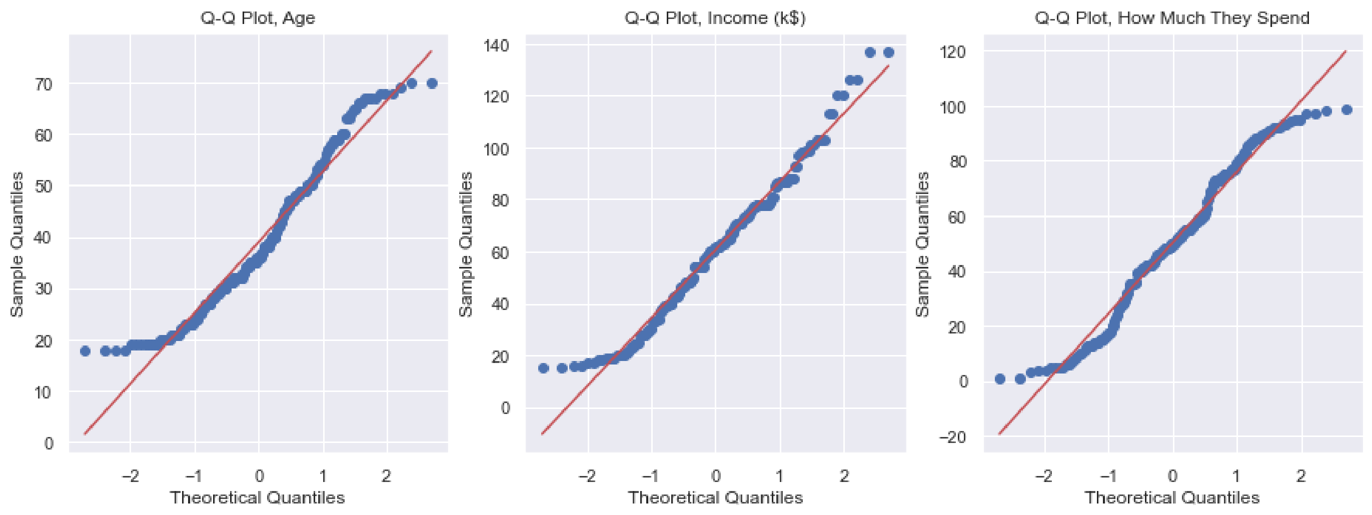
Name: Glenn Wu

Admin No.: 2214395

Class: DAAA/FT/2A/01

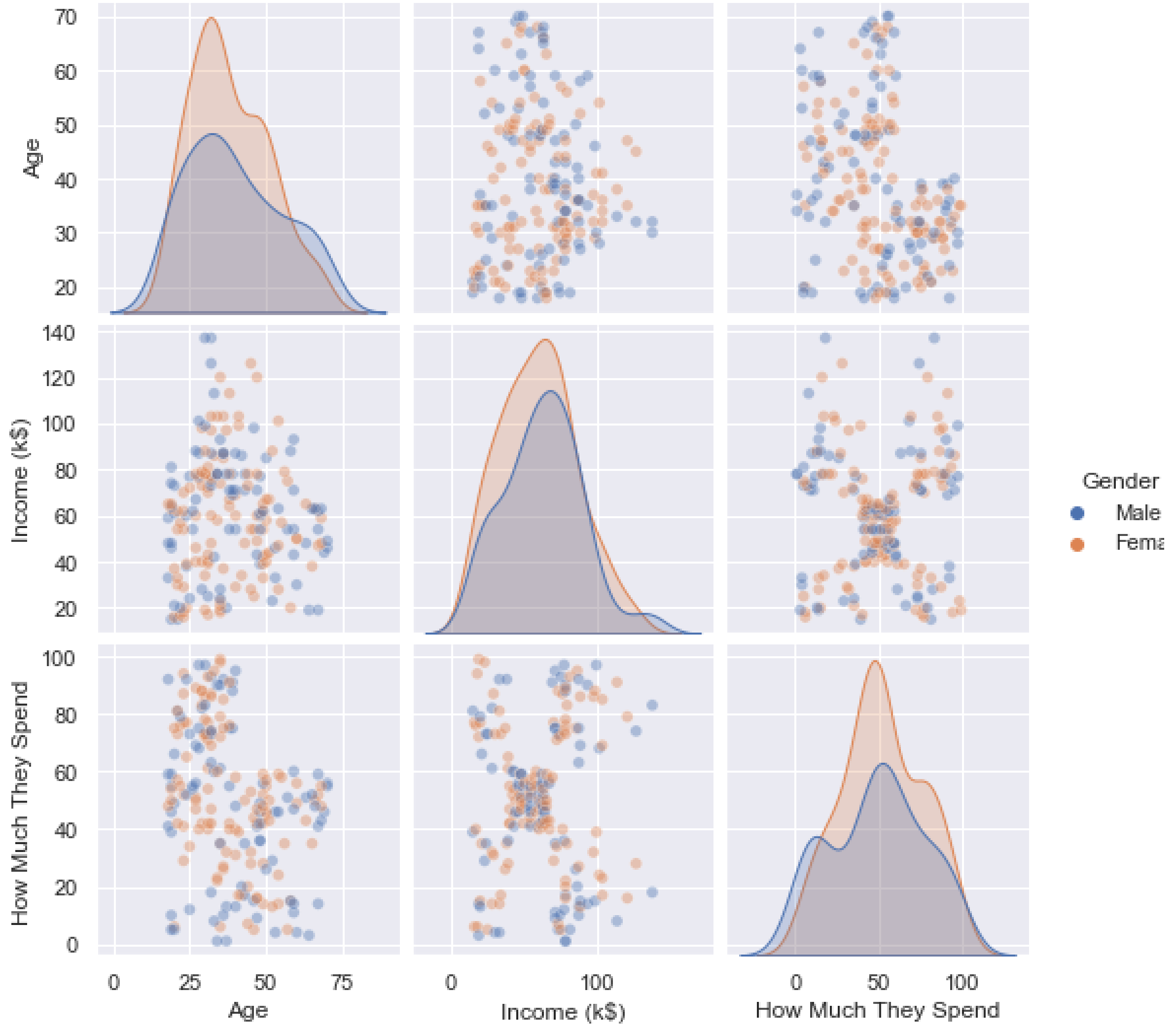
Data Exploration

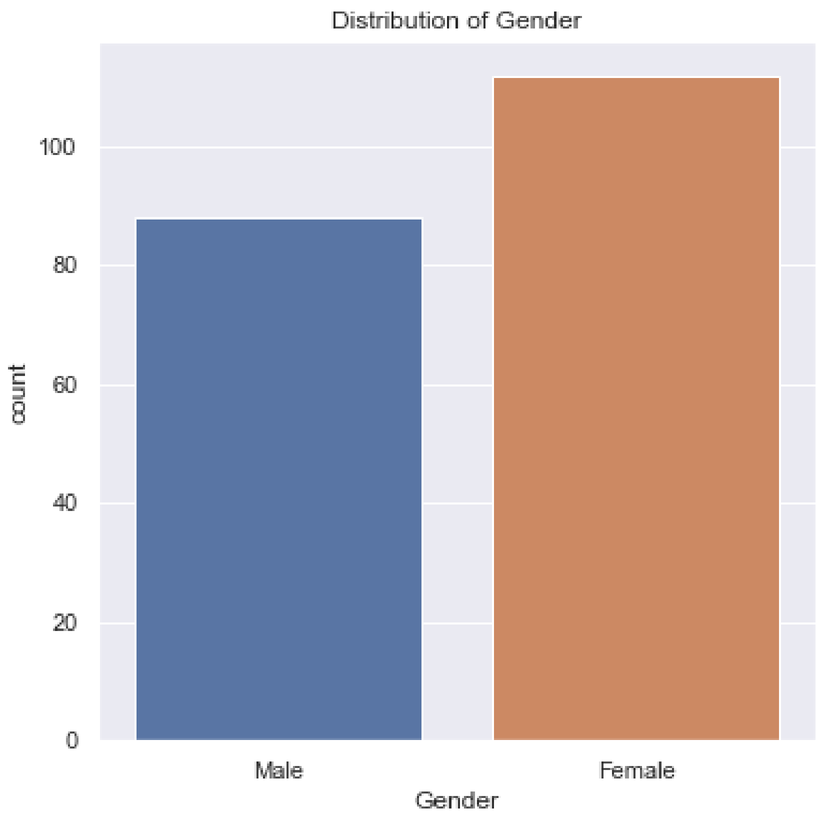




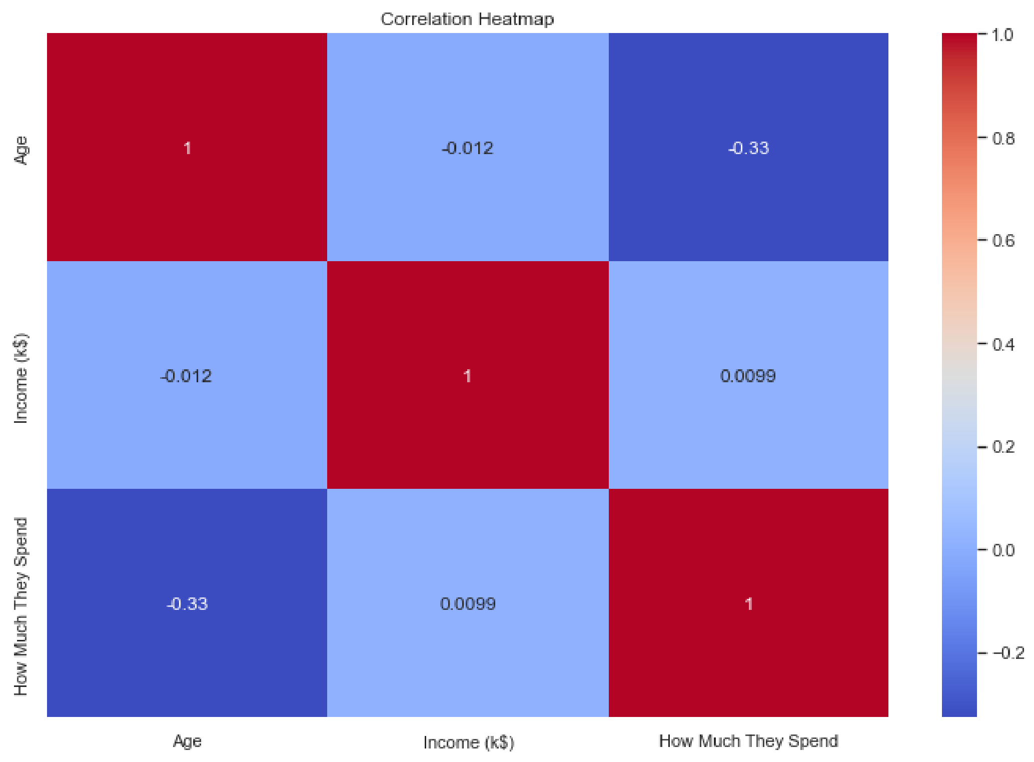
- None of the features follow a gaussian distribution. I will not use Gaussian Mixture Model since that is on of the underlying assumptions.

Data Exploration

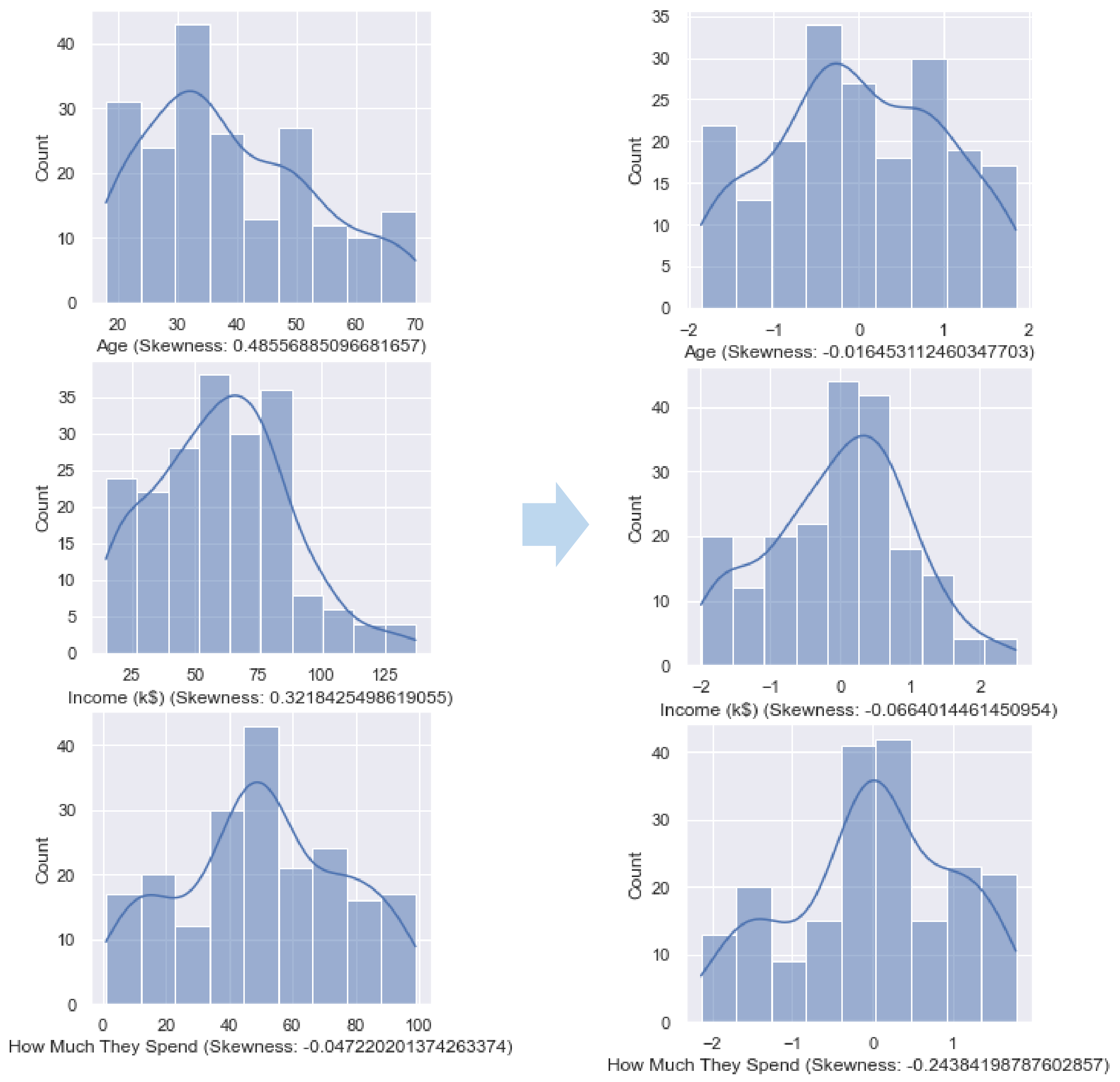




Data Exploration



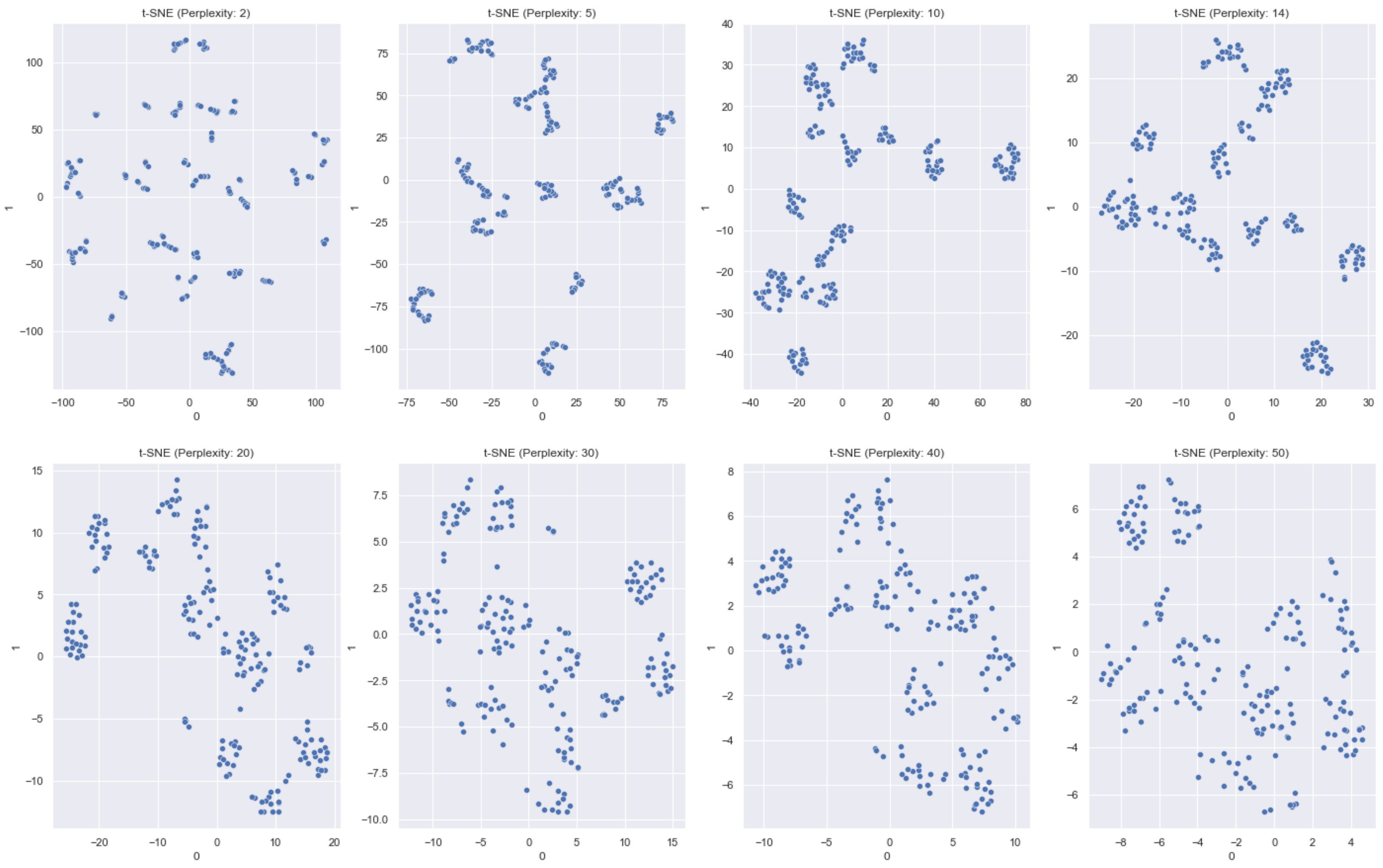
Yeo-Johnson Transformation



Perform this transformation on Age and Income to lower skew

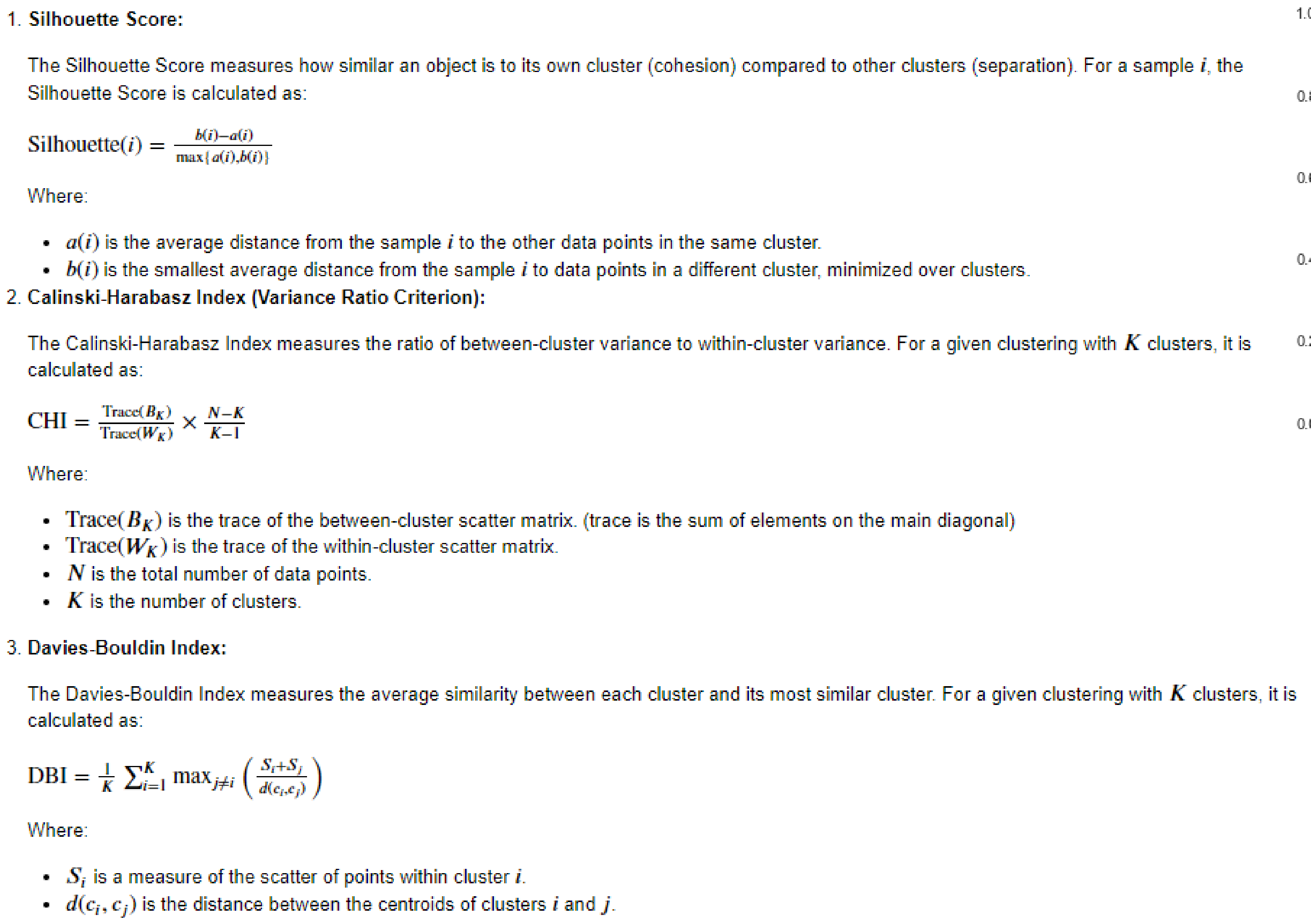
Skew Increased on “How Much They Spend”, only use standaridization.

Data Exploration: t-SNE



- The *perplexity* hyperparameter helps to control how many nearest neighbours each point considers.
- It is useful to consider the t-SNE plot at different *perplexity* values since there is a tradeoff between preserving local and global structures.
- In this case I will consider perplexities of 20 and 30.

Model Selection



1. **Silhouette Score:**

   The Silhouette Score measures how similar an object is to its own cluster (cohesion) compared to other clusters (separation). For a sample i, the Silhouette Score is calculated as:

   $$\text{Silhouette}(i) = \frac{b(i)-a(i)}{\max\{a(i), b(i)\}}$$

   Where:

   - $a(i)$ is the average distance from the sample i to the other data points in the same cluster.
   - $b(i)$ is the smallest average distance from the sample i to data points in a different cluster, minimized over clusters.

2. **Calinski-Harabasz Index (Variance Ratio Criterion):**

   The Calinski-Harabasz Index measures the ratio of between-cluster variance to within-cluster variance. For a given clustering with K clusters, it is calculated as:

   $$\text{CHI} = \frac{\text{Trace}(B_K)}{\text{Trace}(W_K)} \times \frac{N-K}{K-1}$$

   Where:

   - $\text{Trace}(B_K)$ is the trace of the between-cluster scatter matrix. (trace is the sum of elements on the main diagonal)
   - $\text{Trace}(W_K)$ is the trace of the within-cluster scatter matrix.
   - N is the total number of data points.
   - K is the number of clusters.

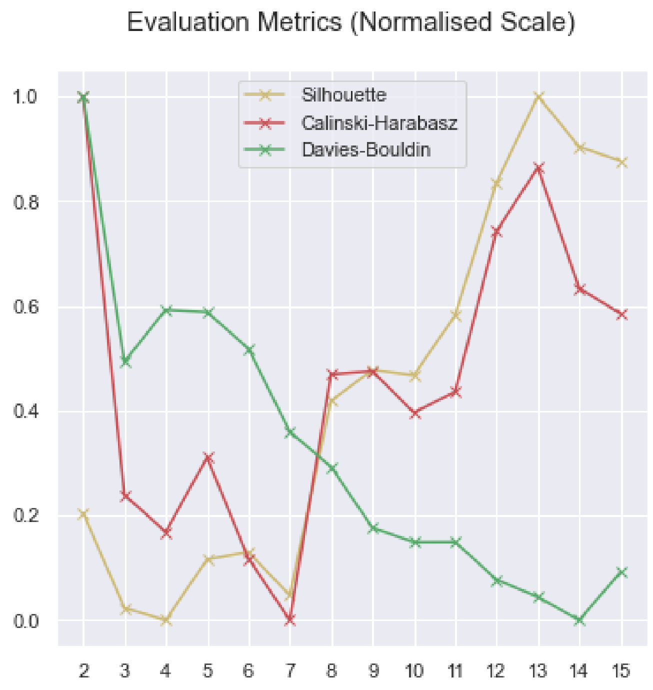
3. **Davies-Bouldin Index:**

   The Davies-Bouldin Index measures the average similarity between each cluster and its most similar cluster. For a given clustering with K clusters, it is calculated as:

   $$\text{DBI} = \frac{1}{K} \sum_{i=1}^{K} \max_{j \neq i} \left(\frac{S_i + S_j}{d(c_i, c_j)} \right)$$

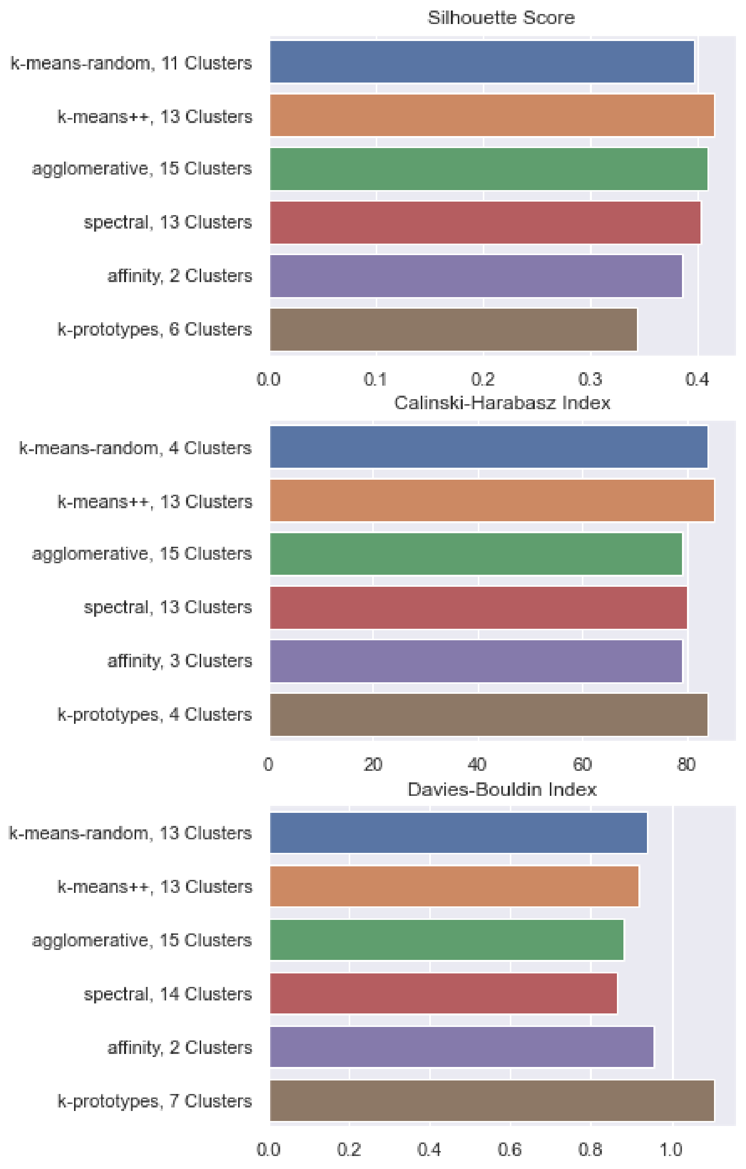
   Where:

   - S_i is a measure of the scatter of points within cluster i.
   - $d(c_i, c_j)$ is the distance between the centroids of clusters i and j.

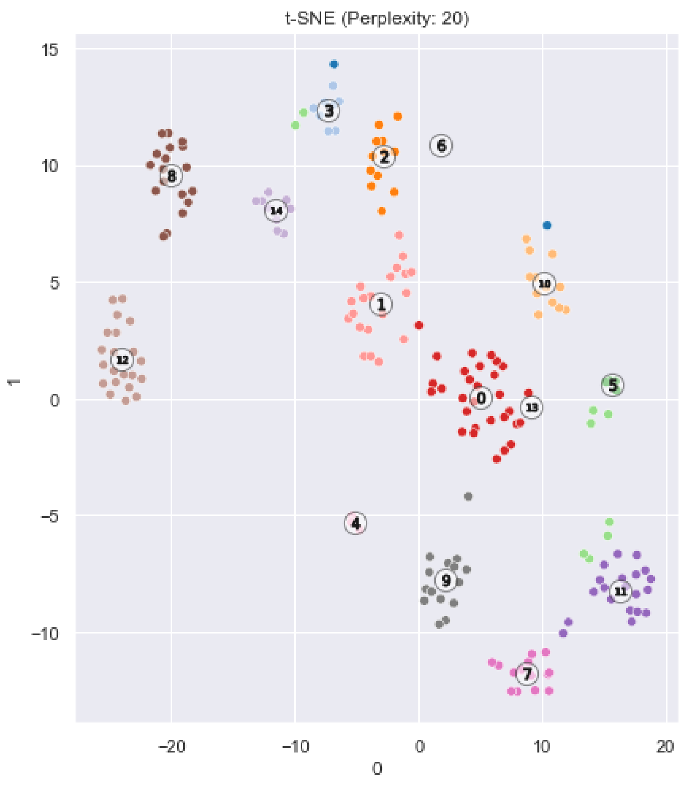


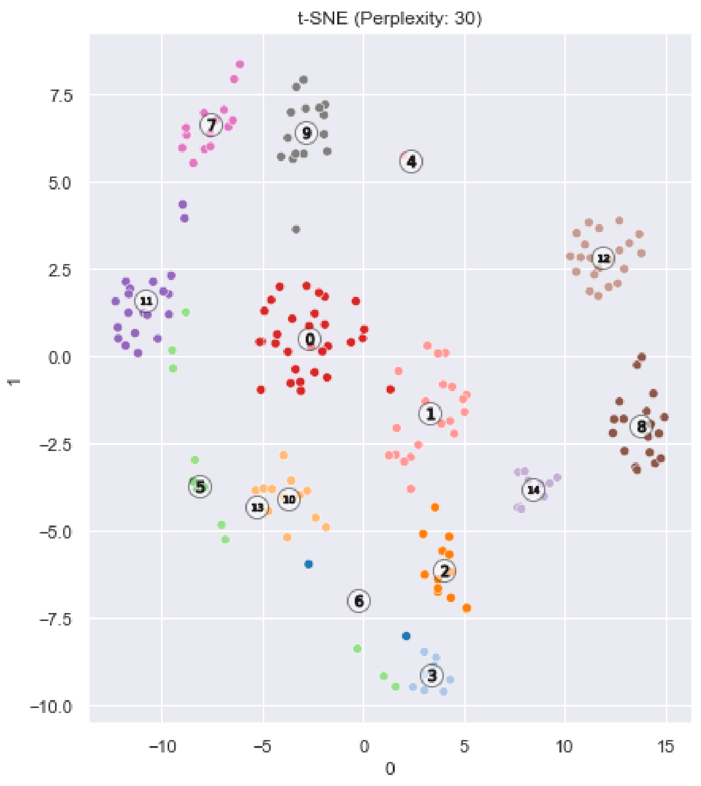
- The metrics are all different methods to measure the quality of the clusters and do not always agree with each other.
- The optimal number of clusters and the best model can be chosen using these metrics.

Model Selection

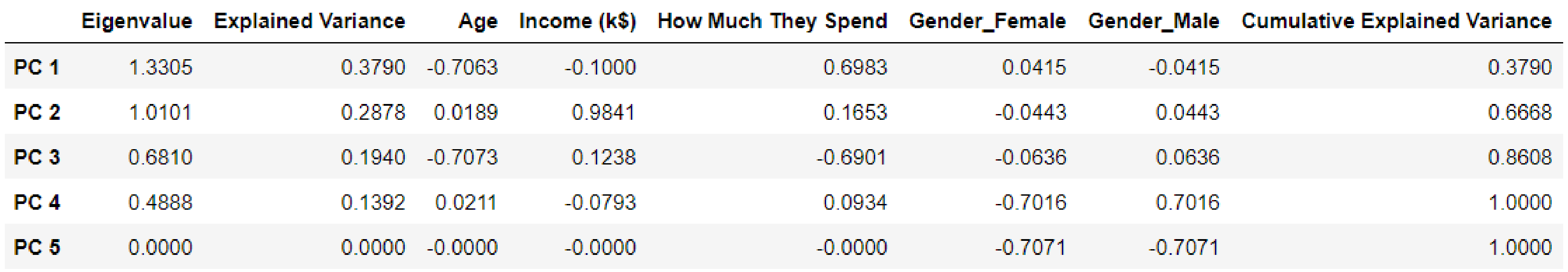


- K-means with k-means++ initialisation seems to perform the best for all metrics at 13 clusters followed by Agglomerative Clustering and K-Prototypes.





Model Improvement: PCA

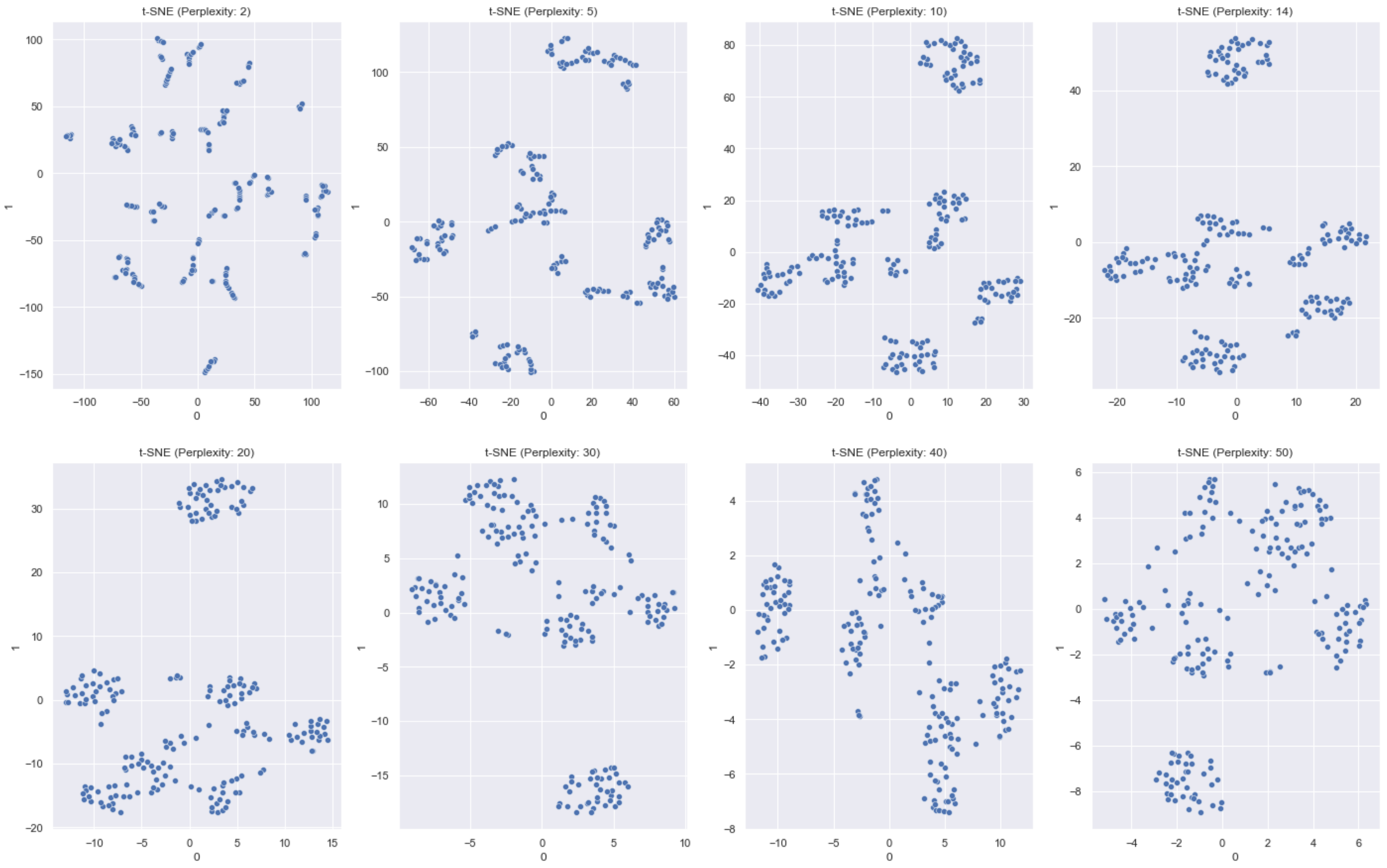


	Eigenvalue	Explained Variance	Age	Income (k$)	How Much They Spend	Gender_Female	Gender_Male	Cumulative Explained Variance
PC 1	1.3305	0.3790	-0.7063	-0.1000	0.6983	0.0415	-0.0415	0.3790
PC 2	1.0101	0.2878	0.0189	0.9841	0.1653	-0.0443	0.0443	0.6668
PC 3	0.6810	0.1940	-0.7073	0.1238	-0.6901	-0.0636	0.0636	0.8608
PC 4	0.4888	0.1392	0.0211	-0.0793	0.0934	-0.7016	0.7016	1.0000
PC 5	0.0000	0.0000	-0.0000	-0.0000	-0.0000	-0.7071	-0.7071	1.0000

- Only PC 5 explains less than 10% of total variance (Kaiser's Rule)
- To obtain more than 80% cumulative explained variance, we have to select 3 PCs
- The main weights of PC4 and PC5 and on Gender_Female and Gender_Male.

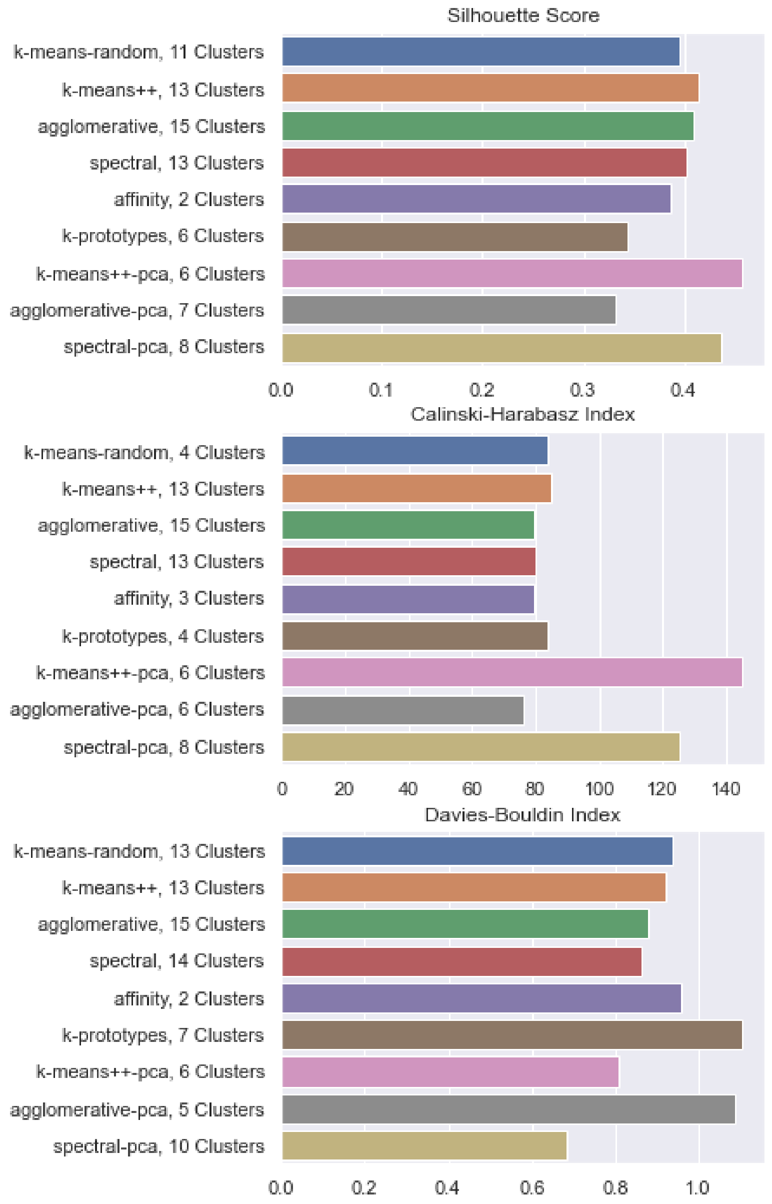
- One of the features from the transformed data is completely redundant, either Gender_Male or Gender_Female, since One-Hot encoding makes creates a column for each unique value in the feature. So minimally we should expect that we can discard the last Principal Component.
- Using only Prinicipal Components that have a Cumulative Explained Variance of 80%, we can also discard the 4th PC as well.
- Thus, I will use PC 1, 2, and 3.

Model Improvement: PCA

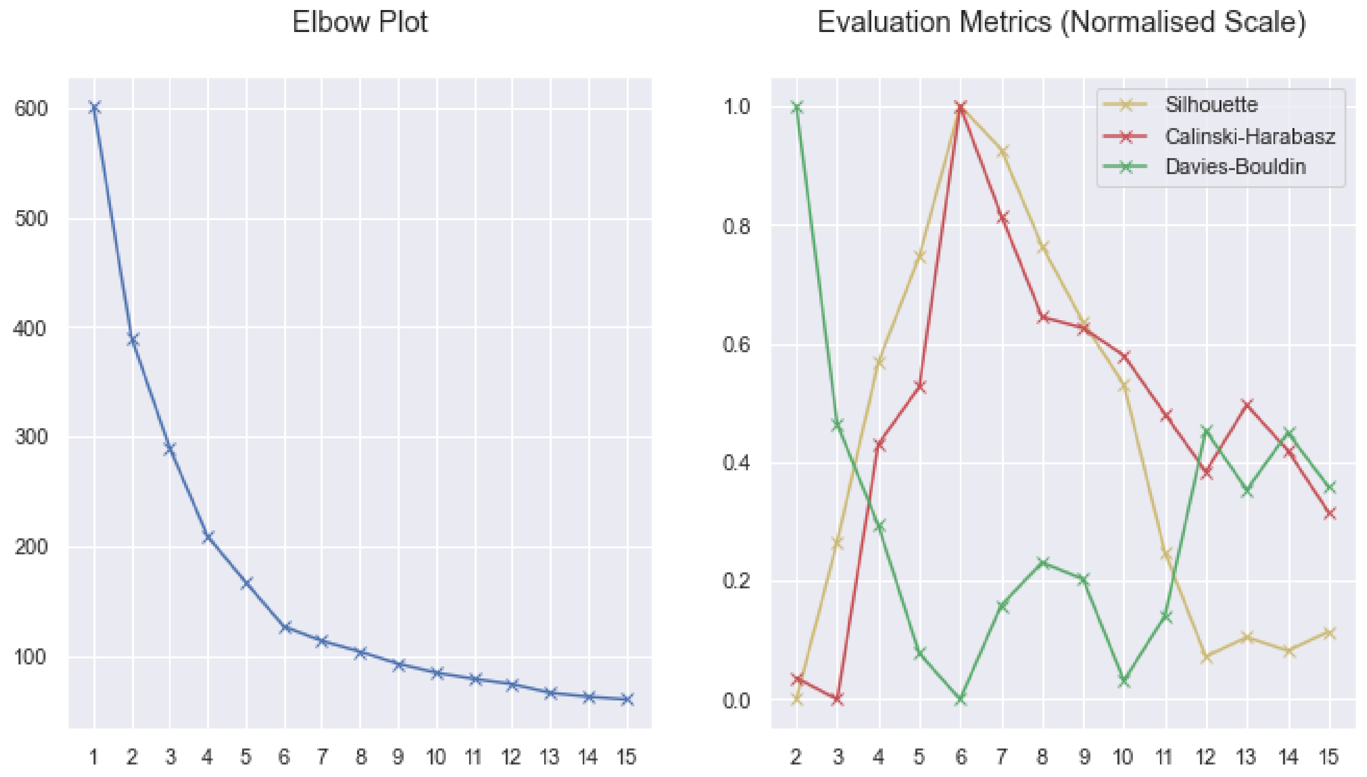


- In this case I will consider perplexities of 14 and 30.

Model Improvement: PCA



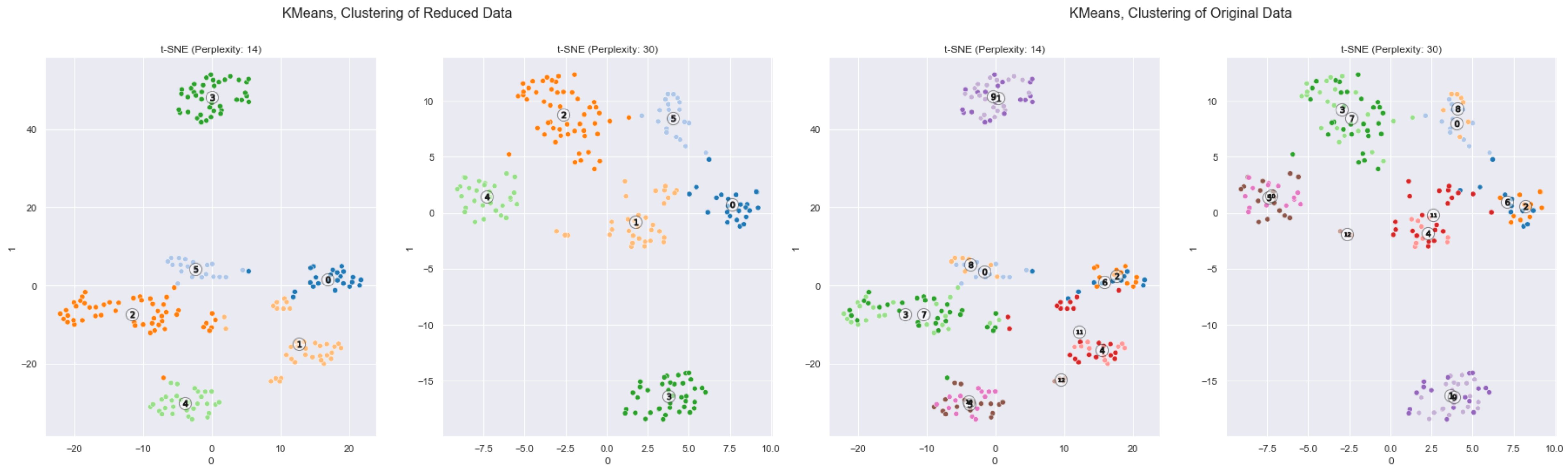
- K-means with k-means++ initialisation performs the best across all metrics at 6 clusters.



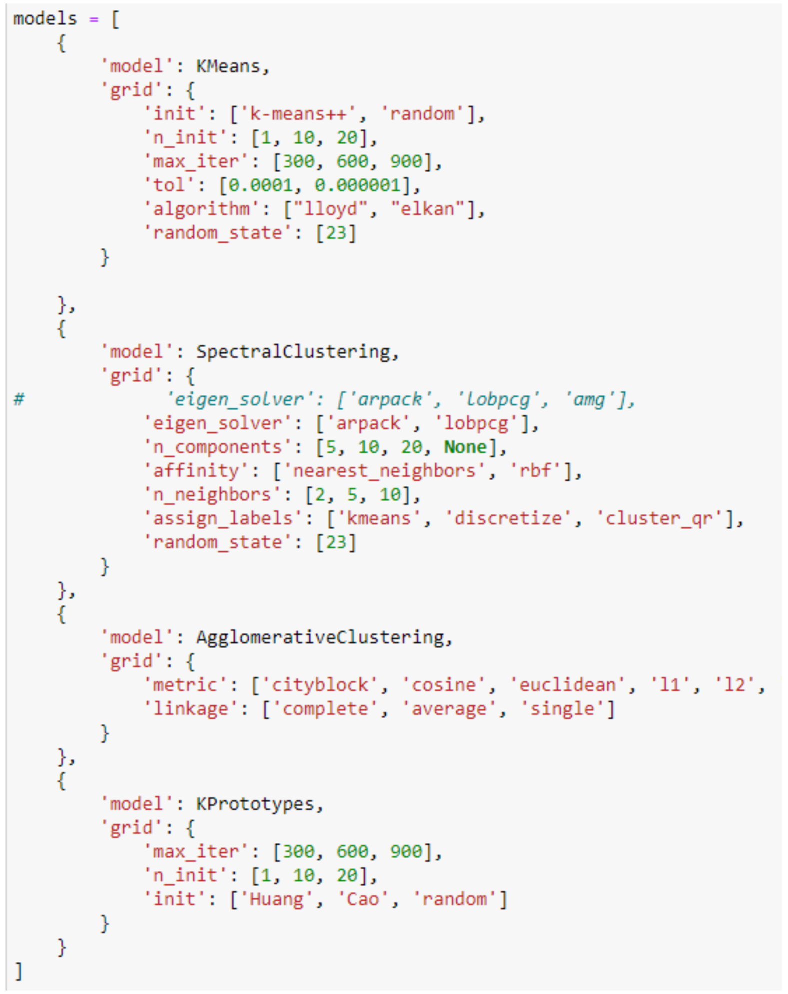
K-Means

Model Improvement: PCA

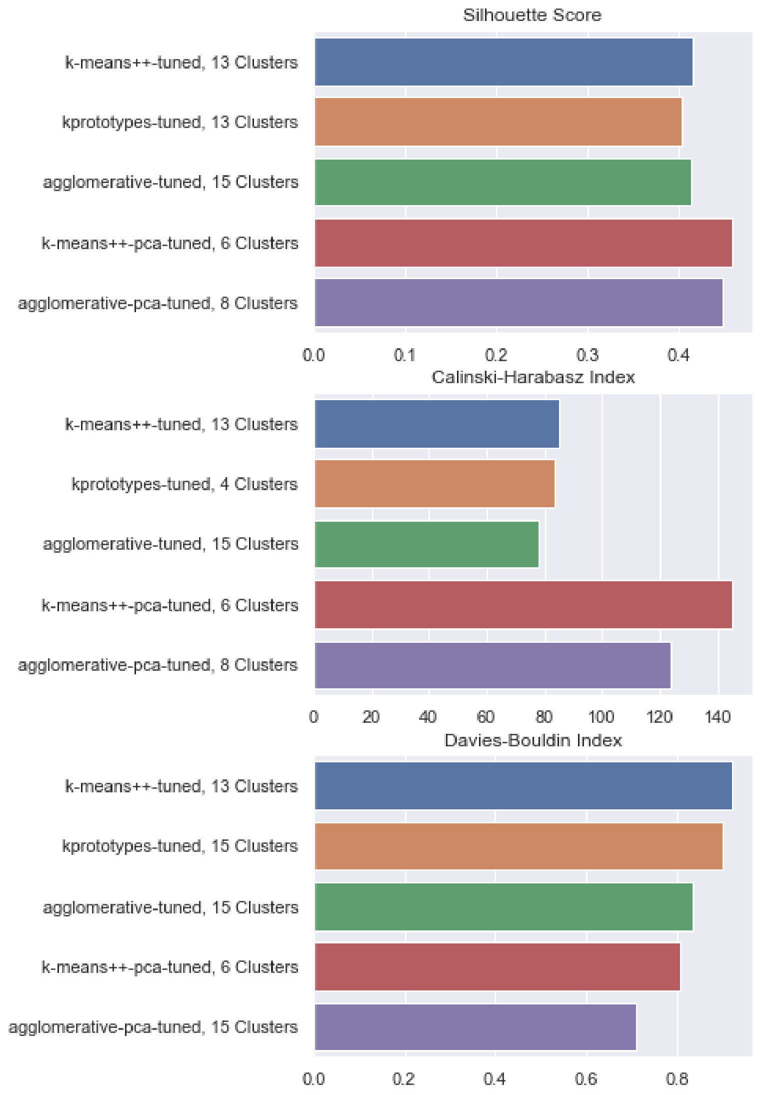
- PCA effectively removed the Gender columns since they contributed the least to the overall explained variance.
- E.g. it combined the 13 clusters from K-Means on the original data to 6 clusters in the reduced data



Model Improvement: Hyperparameter Tuning

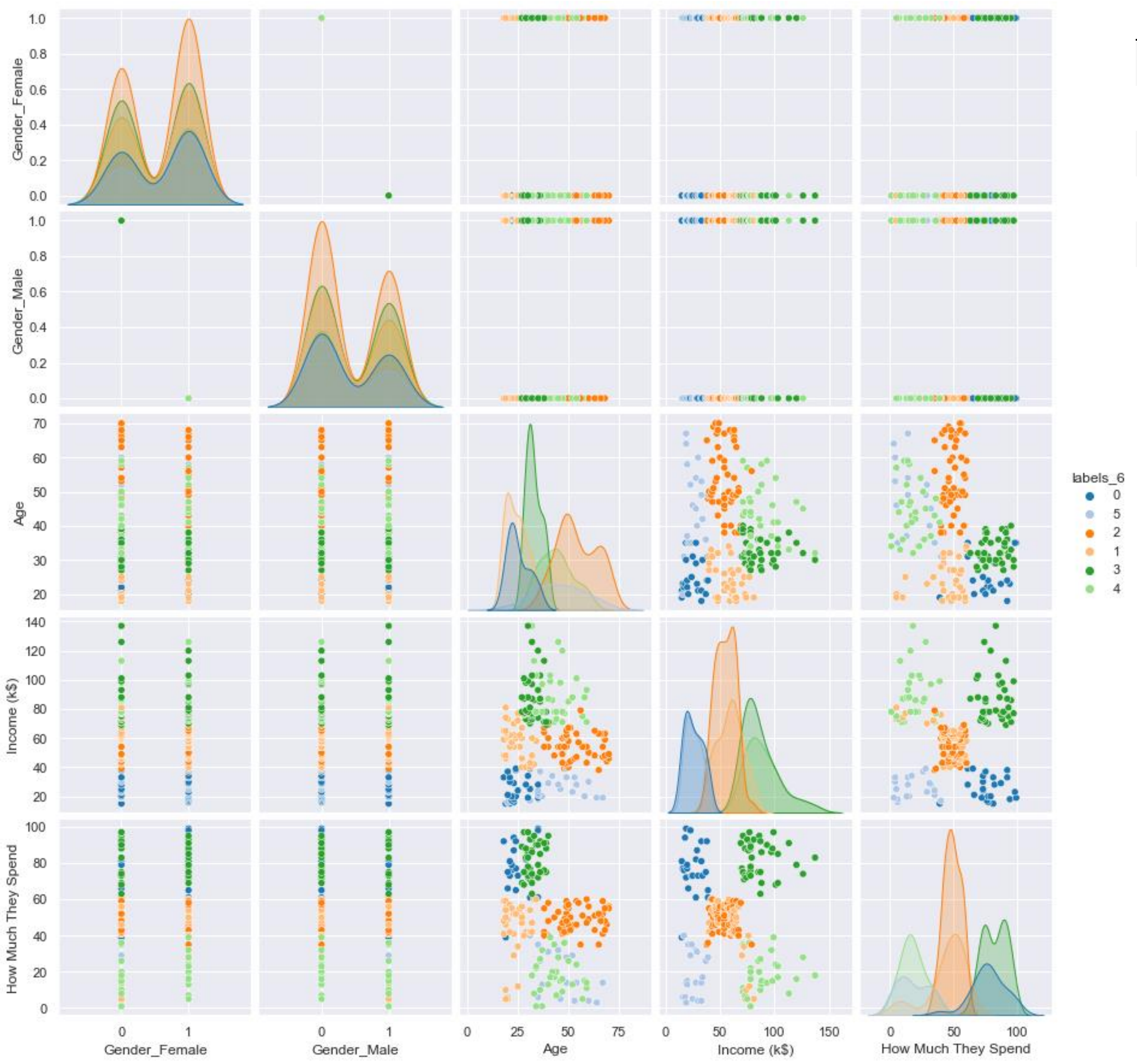


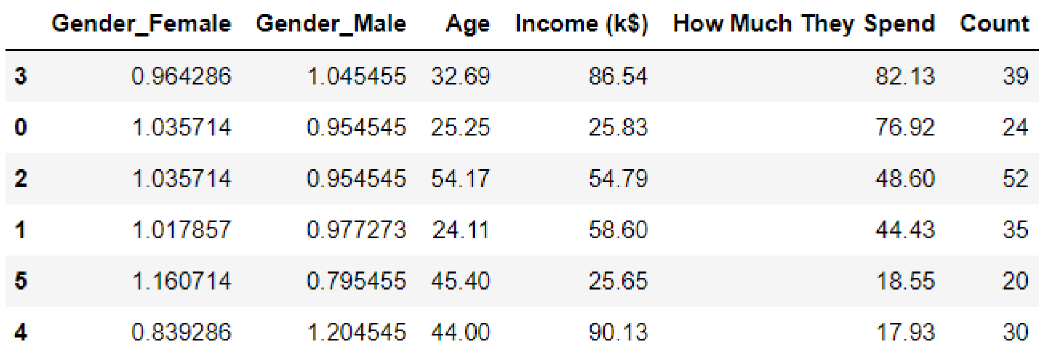
```
models = [
    {
        'model': KMeans,
        'grid': {
            'init': ['k-means++', 'random'],
            'n_init': [1, 10, 20],
            'max_iter': [300, 600, 900],
            'tol': [0.0001, 0.000001],
            'algorithm': ["lloyd", "elkan"],
            'random_state': [23]
        }

    },
    {
        'model': SpectralClustering,
        'grid': {
#               'eigen_solver': ['arpack', 'lobpcg', 'amg'],
            'eigen_solver': ['arpack', 'lobpcg'],
            'n_components': [5, 10, 20, None],
            'affinity': ['nearest_neighbors', 'rbf'],
            'n_neighbors': [2, 5, 10],
            'assign_labels': ['kmeans', 'discretize', 'cluster_qr'],
            'random_state': [23]
        }
    },
    {
        'model': AgglomerativeClustering,
        'grid': {
            'metric': ['cityblock', 'cosine', 'euclidean', 'l1', 'l2',
            'linkage': ['complete', 'average', 'single']
        }
    },
    {
        'model': KPrototypes,
        'grid': {
            'max_iter': [300, 600, 900],
            'n_init': [1, 10, 20],
            'init': ['Huang', 'Cao', 'random']
        }
    }
]
```



- The only significant improvement is for Agglomerative Clustering on PCA with a improving Silhouette score of 0.330898 to 0.447471
- K-Means is still the best performing and its Silhouette score remained unchanged at 0.457448

Conclusion: Cluster Interpretations (K-Means)





	Gender_Female	Gender_Male	Age	Income (k$)	How Much They Spend	Count
3	0.964286	1.045455	32.69	86.54	82.13	39
0	1.035714	0.954545	25.25	25.83	76.92	24
2	1.035714	0.954545	54.17	54.79	48.60	52
1	1.017857	0.977273	24.11	58.60	44.43	35
5	1.160714	0.795455	45.40	25.65	18.55	20
4	0.839286	1.204545	44.00	90.13	17.93	30

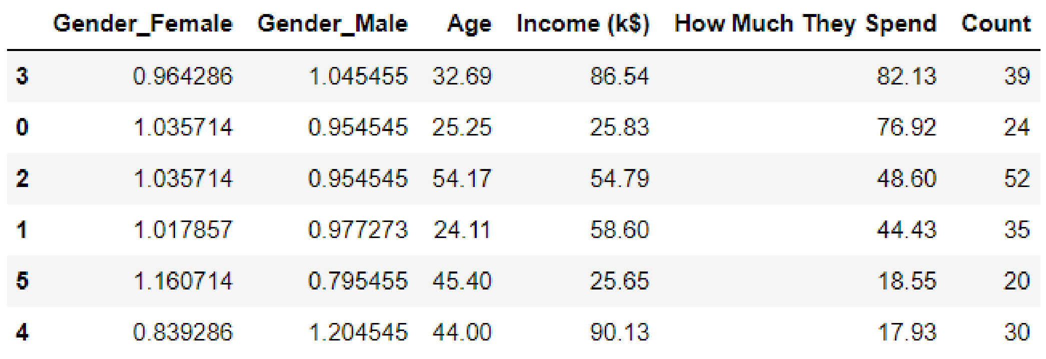
Cluster 0: "Young Spenders"

Demographic: Young (Average Age: 25.25), Low to Moderate Income (25.83 k$), High Spending (76.92)
Interpretation: This cluster represents young individuals with relatively lower income who spend a significant portion of their income. They might be early in their careers or education and prioritize spending on various products.

Cluster 1: "Economical Shoppers"

Demographic: Young (Average Age: 24.11), Moderate Income (58.60 k$), Moderate Spending (44.43)
Interpretation: This cluster consists of individuals who are relatively young and have moderate income. While they do spend, they tend to be more cautious and economical in their spending habits.

Conclusion: Cluster Interpretations (K-Means)



	Gender_Female	Gender_Male	Age	Income (k$)	How Much They Spend	Count
3	0.964286	1.045455	32.69	86.54	82.13	39
0	1.035714	0.954545	25.25	25.83	76.92	24
2	1.035714	0.954545	54.17	54.79	48.60	52
1	1.017857	0.977273	24.11	58.60	44.43	35
5	1.160714	0.795455	45.40	25.65	18.55	20
4	0.839286	1.204545	44.00	90.13	17.93	30

Cluster 2: "Mid-Age Budgeters"

Demographic: Mid-Age (Average Age: 54.17), Moderate Income (54.79 k$), Moderate Spending (48.60)
Interpretation: This cluster includes individuals in their mid-age years who have moderate income and spending patterns. They may have established financial responsibilities and make reasonable but not excessive purchases.

Cluster 3: "Affluent Shoppers"

Demographic: Young (Average Age: 32.69), High Income (86.54 k$), Very High Spending (82.13)
Interpretation: This cluster represents affluent individuals, both young and mid-age, who have high incomes and consequently high spending patterns. They likely prioritize quality and luxury in their purchases.

Cluster 4: "High-Income Savers"

Demographic: Mid-Age (Average Age: 44.00), High Income (90.13 k$), Low Spending (17.93)
Interpretation: This cluster consists of mid-age individuals, moreso males, with high incomes but relatively low spending patterns. They might be more conservative in their spending and focus on savings or specific investments.

Cluster 5: "Frugal Shoppers"

Demographic: Mid-Age (Average Age: 45.40), Low Income (25.65 k$), Very Low Spending (18.55)
Interpretation: This cluster represents individuals, especially females, with mid-age and relatively lower incomes. They exhibit frugal spending habits and are careful with their purchases.

Conclusion: Cluster Interpretations (Agglomerative)

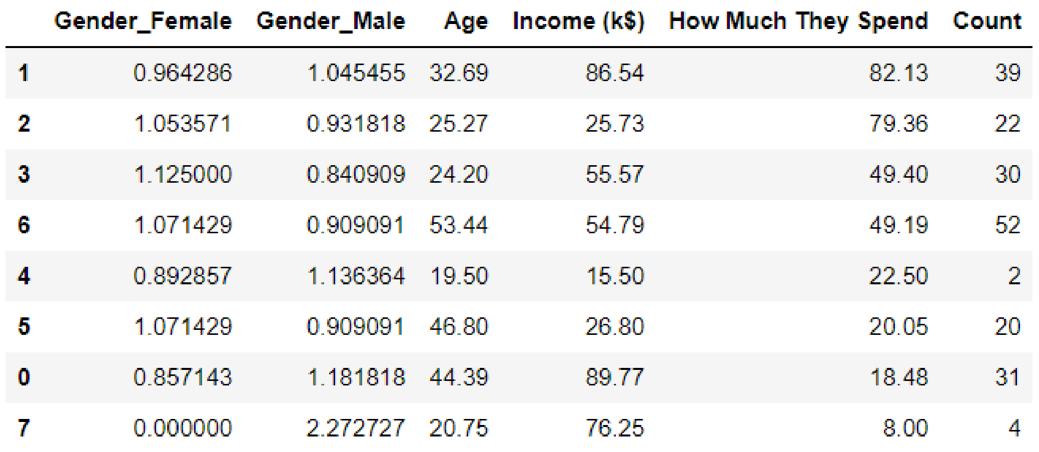
The same 6 clusters from K-Means are featured with nearly identical demographics but with an additional 2 clusters.

Cluster 4: "Young Frugal Shoppers“

Demographic: Very Young (Average Age: 19.50), Very Low Income (15.50 k$), Low Spending (22.50) Interpretation: This cluster consists primarily of very young individuals, mostly males, with very low income and relatively conservative spending patterns. They exhibit frugal behaviors, likely due to financial constraints or early financial independence.

Cluster 7: "Outlier Spenders“

Demographic: Primarily Males, Very Young (Average Age: 20.75), High Income (76.25 k$), Very High Spending (8.00) Interpretation: This cluster consists exclusively of very young males with high income and extremely high spending patterns. They appear to be outliers with exceptional spending behaviors that deviate significantly from the rest of the groups.



	Gender_Female	Gender_Male	Age	Income (k$)	How Much They Spend	Count
1	0.964286	1.045455	32.69	86.54	82.13	39
2	1.053571	0.931818	25.27	25.73	79.36	22
3	1.125000	0.840909	24.20	55.57	49.40	30
6	1.071429	0.909091	53.44	54.79	49.19	52
4	0.892857	1.136364	19.50	15.50	22.50	2
5	1.071429	0.909091	46.80	26.80	20.05	20
0	0.857143	1.181818	44.39	89.77	18.48	31
7	0.000000	2.272727	20.75	76.25	8.00	4