Car Recommender

Omar Qusous

https://github.com/oqusous/car_recommendation_

Content based Recommender for cars

Data Sources

https://www.carspecs.us -BS4

https://www.kbb.com/ - Selenium

All models and trims of 43 Car brands - 2234 in total

```
In [30]:  1  df_gas_mod.index[:20]

Out[30]: Index(['/cars/2019/acura/ilx/79088', '/cars/2019/acura/ilx/79090',
       '/cars/2019/acura/ilx/79089', '/cars/2019/acura/ilx/79092',
       '/cars/2019/acura/ilx/79091', '/cars/2019/acura/mdx/77624',
       '/cars/2019/acura/mdx/77621', '/cars/2019/acura/mdx/77620',
       '/cars/2019/acura/mdx/77623', '/cars/2019/acura/mdx/77622',
       '/cars/2019/acura/mdx/77619', '/cars/2019/acura/mdx/77625',
       '/cars/2019/acura/mdx/77626', '/cars/2019/acura/mdx/77628',
       '/cars/2019/acura/mdx/77627', '/cars/2019/acura/mdx/77631',
       '/cars/2019/acura/mdx/77629', '/cars/2019/acura/mdx/77630',
       '/cars/2019/acura/nsx/79093', '/cars/2019/acura/rdx/77635'],
      dtype='object')
```

5 trims of Acura ILX model

12 trims of Acura MDX model

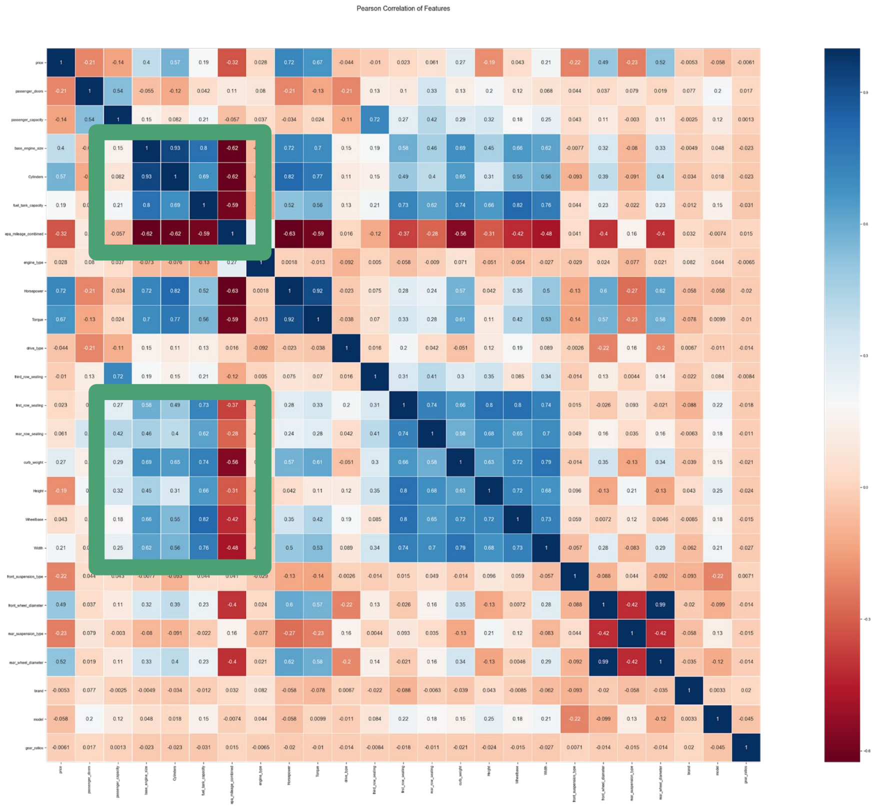
Features

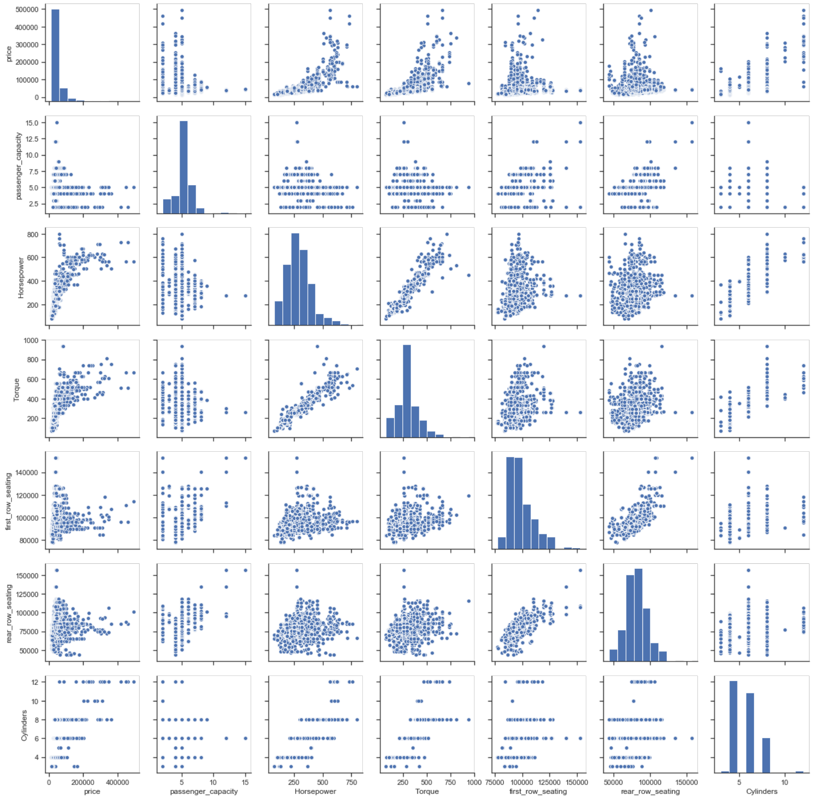
- Physical: external and internal dimensions and weight of the car
- Performance: Horsepower, Cylinders, Torque and Gear Ratios
- Value: Price
- Engine type: Fuel Capacity, Hybrid or Gas and miles/gallon
- Total- 35 features

Feature Engineering

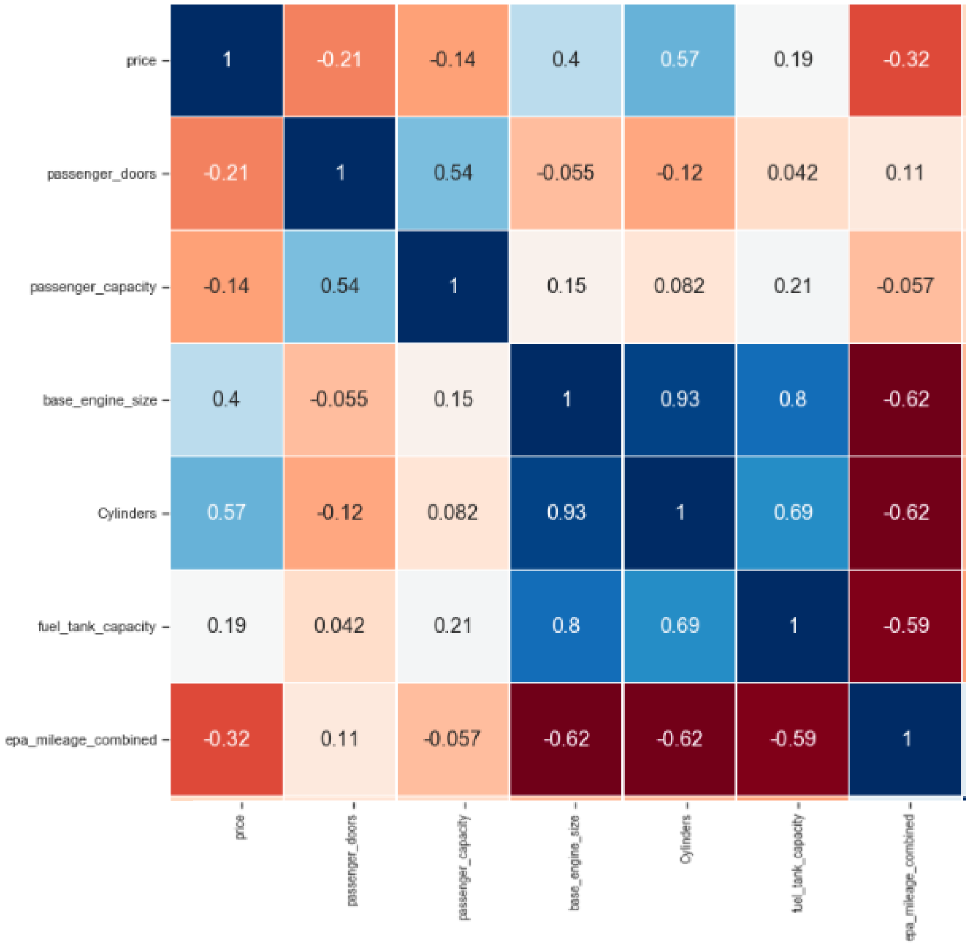
- Domain knowledge selection
- Continuous Features:
  - MinMaxScaler
  - Feature interaction: interior space and gear ratios
- Categorical
  - Dummies for passenger capacity, engine type, number of cylinders, etc..
- Missing data mainly handled using KNN Classifier and Regressor

EDA 1- Corr Heatmap and Feat. Pair Plots





EDA 2



Price

No. Doors

Pass Cap.

Engine size

Cylinders

Fuel tank

mile/gallon

EDA 3

Front seats
Second seats
Weight
Height
Length
Width

Price
No. Doors
Pass Cap.
Engine size
Cylinders
Fuel tank
mile/gallon

Modelling

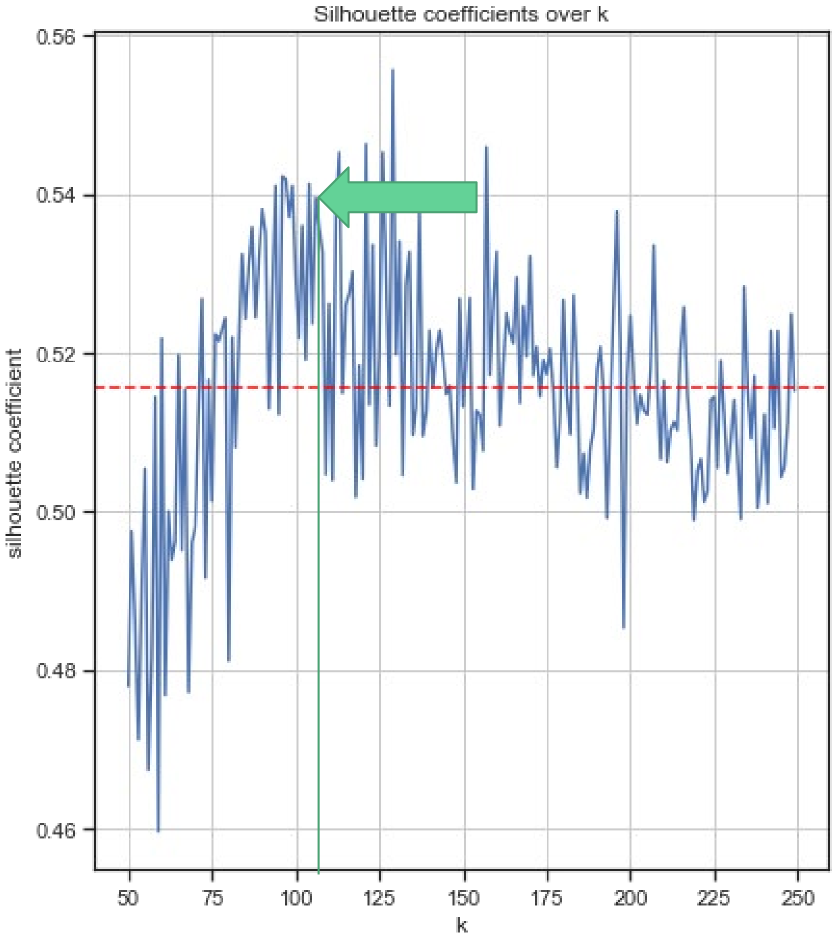
Clustering:

- KMeans
- KAgglomerativeClustering
- Annoy - Spotify made

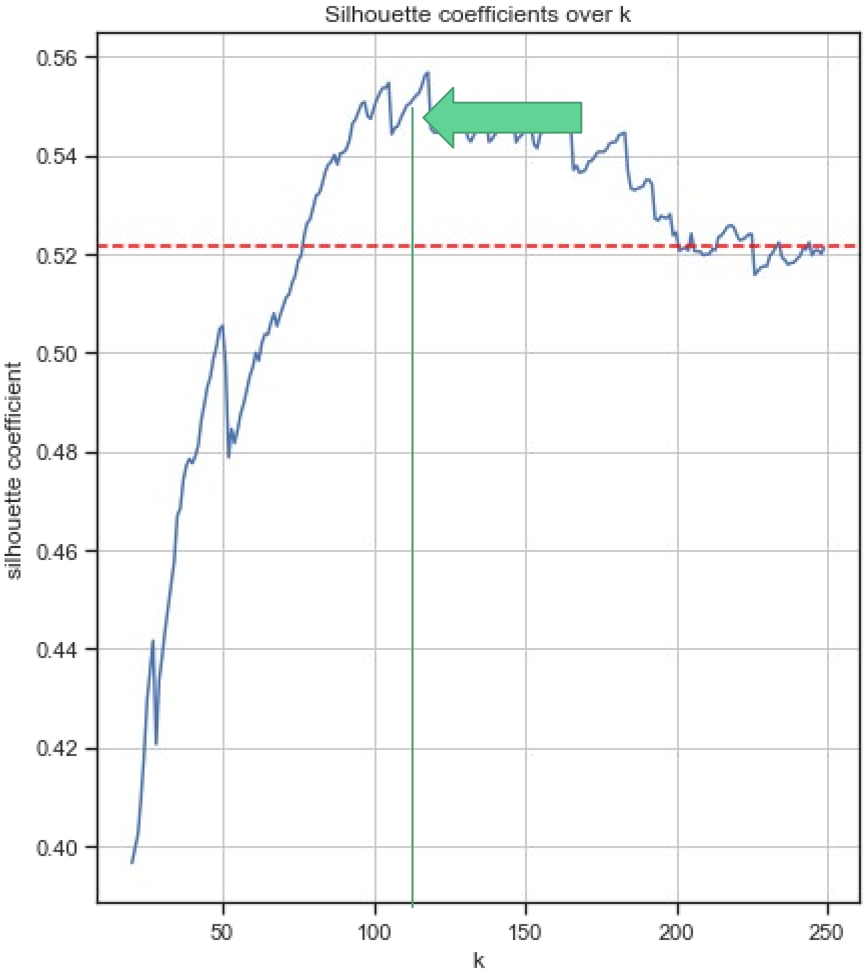
KAgglomerativeClustering v Kmeans

- Clusters, k, varied between 20-250; k=110 gave best results.
- Ward linkage in KAgglomerativeClustering parameter outperformed the rest of the available methods.
- Both were run with and without PCA.

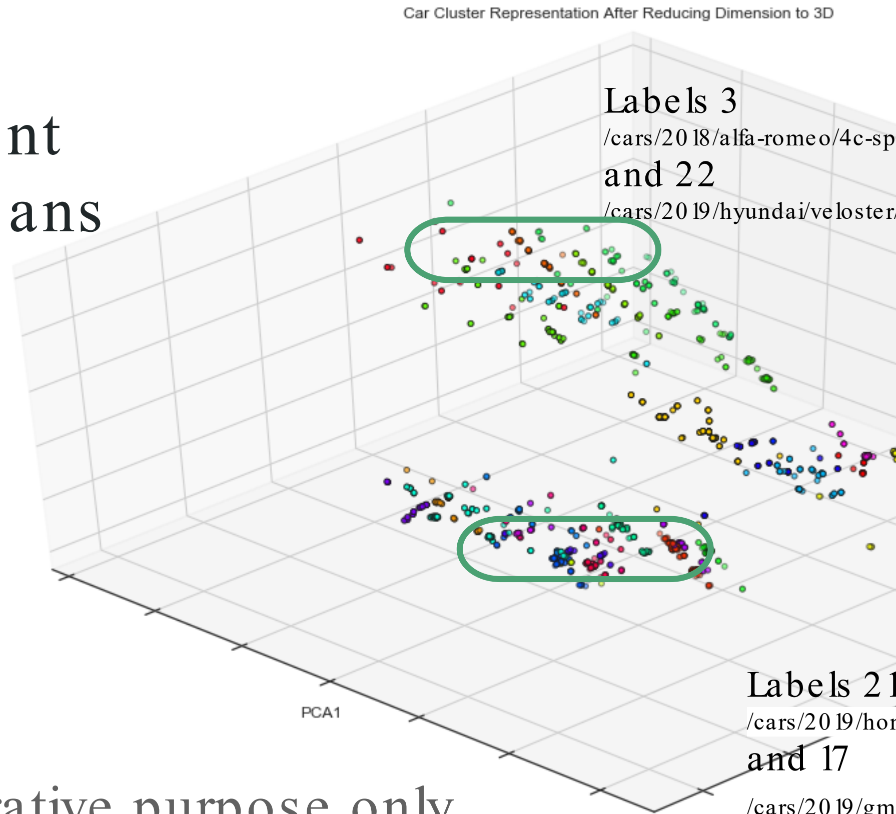
KMeans Silhouette Plot



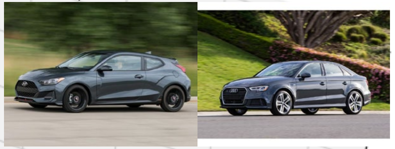
KAgglomerati veClustering Silhouette Plot



Three component PCA KMeans plot



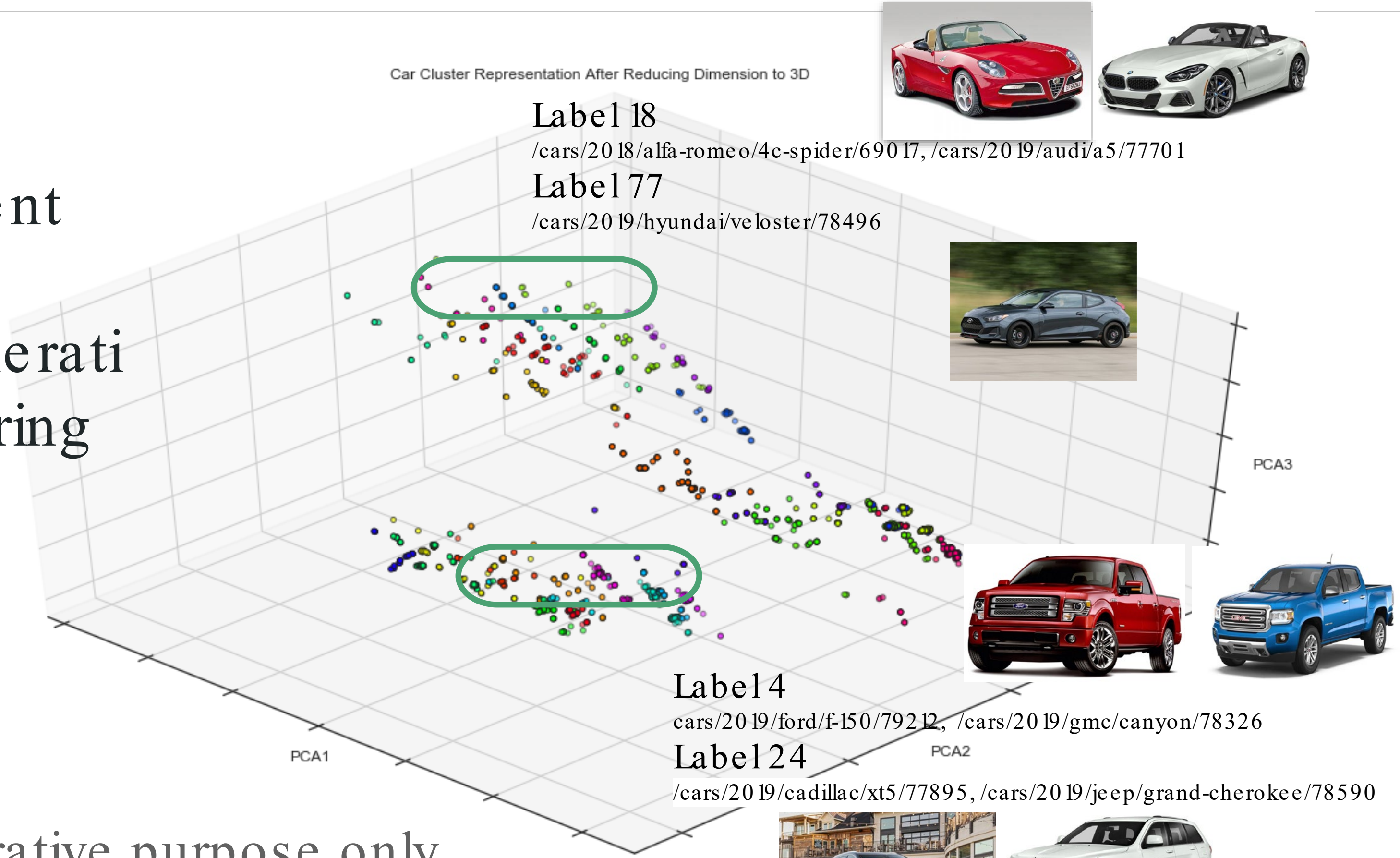
Labels 3
/cars/2018/alfa-romeo/4c-spider/69017, /cars/2019/bmw/z4/79154
and 22
/cars/2019/hyundai/veloster/78496, /cars/2019/audi/a3-cabriolet/77681



Labels 21
/cars/2019/honda/passport/79308,, /cars/2019/toyota/avalon/78960
and 17
/cars/2019/gmc/canyon/78326, /cars/2019/ford/f-150/79212

*For illustrative purpose only

Three component PCA KAgglomerativeClustering plot



Label 18

/cars/2018/alfa-romeo/4c-spider/69017, /cars/2019/audi/a5/77701

Label 77

/cars/2019/hyundai/veloster/78496

Label 4

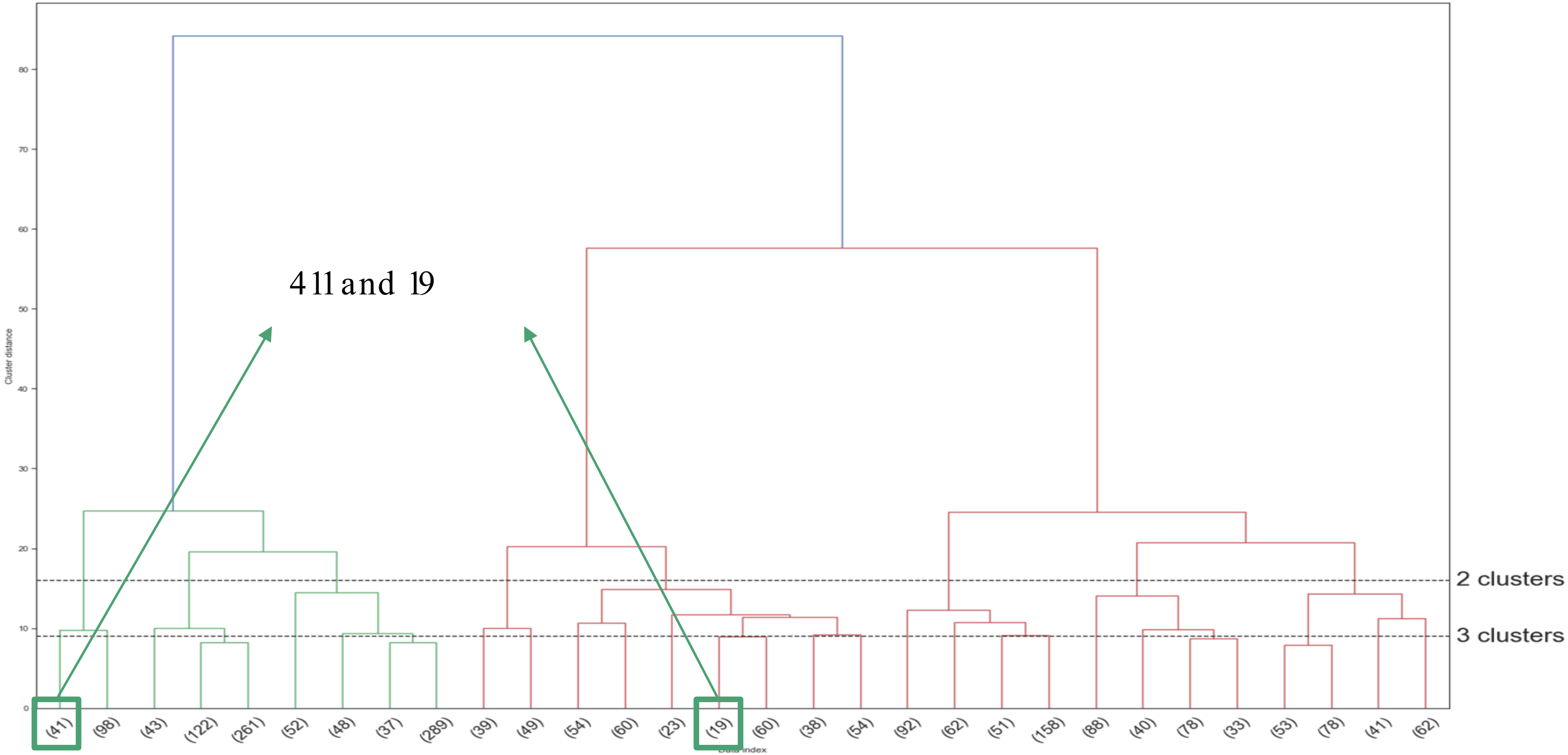
cars/2019/ford/f-150/79212, /cars/2019/gmc/canyon/78326

Label 24

/cars/2019/cadillac/xt5/77895, /cars/2019/jeep/grand-cherokee/78590


*For illustrative purpose only

KAgglomerativeClustering Ward Dendogram



KMeans v KAgglomerativeClustering

Model	calinski_harabasz_score	silhouette_score
KMeans (k=30,)	642	0.43
KAgglomerativeClustering (k=110, linkage = single)	289	0.54
KAgglomerativeClustering (k=110,linkage = ward)	**811**	**0.55**
KAgglomerativeClustering (k=110,linkage = average)	428	0.58
KAgglomerativeClustering (k=110,linkage = complete)	475	0.53
PCA 15 Components with KAgglomerativeClustering	730	0.48

Annoy (Approximate Nearest Neighbors Oh Yeah)

https://github.com/spotify/annoy

Annoy is a C++ library with Python bindings to search for points in space that are close to a given query point. It also creates large read-only file-based data structures that are mmapped into memory so that many processes may share the same data.

It achieved the best results

Annoy + Streamlit Script

Vectorize features

```
from sklearn.decomposition import TruncatedSVD
svd = TruncatedSVD(n_components=11)
df_Annoy_svd = svd.fit_transform(df_gas_mod)
print(np.cumsum(svd.explained_variance_ratio_))
[0.40968461 0.61621028 0.69876376 0.75899439 0.81644876 0.85216069
 0.88574629 0.91152626 0.92917066 0.94224498 0.95301177]
```

builds a forest of `n_trees` trees. More trees gives higher precision. `AnnoyIndex(f, metric)` returns a new index that's read-write and stores vector of `f` dimension

```
from annoy import AnnoyIndex
f = df_Annoy_svd.shape[1] # Length of item vector that will be indexed
t = AnnoyIndex(f)
for i in range(df_Annoy_svd.shape[0]):
    v = df_Annoy_svd[i]
    t.add_item(i, v)
t.build(15)
t.save('annoy_svd.ann')
```

Function to find nn by index

```
def nearest_car_Annoy(df, car_idx, index, n):
    nn = index.get_nns_by_item(car_idx, n)
    print('Closest to %s : \n' % df.index[car_idx])
    cars = [df.index[i] for i in nn]
    return df_for_brands_gas.loc[cars, ['brand','model','Torque','senger Capacity',
'price', 'trim']]
```

What is next?

- Integrate electric cars
- 2020 models
- Find other data sources to scrape for cars that required more KNN predictions
- Hybrid recommender with user and expert ratings
- Formulate value for options like infotainment system and interior material quality.