오늘도 어김없이 자료 공유의 날이 찾아왔습니다. 언제 올라오나 발을 동동 구르며 애타게 기다리시던 분들이 눈에 선한데요. 늦어서 죄송합니다.

전날 대방출했던 꿀팁은 설 연휴에 잘 사용하셨나요. 그 팁을 깨달았던 날 머릿 속에 폭죽이 터지는 듯한 흥분을 가라앉히지 못하고 잠을 뒤척였던 때가 아직도 생생합니다.

오늘은 피카츄와 함께하는 마지막 날입니다. 이렇게 쓰니 벌써 가슴 한켠이 아려옵니다.
기억나시나요? 피카츄와 함께 어떻게 볼에 전기 모으는지도 보고, 또 모은 전기를 쏘아보기도 했었쥬. 이제 체육관을 정복 아니 파이리와 꼬부기를 만나러 아니 우리 회사에도 파이리, 꼬부기 닮은 분이 아니 그게 아니라. 정신 차리겠습니다.
지금은 PCA 편으로 단순히 사용가능한 대시보드 하나가 추가되는 느낌이지만 이후 함께하실 classification+probability prediction, regression, time-series forecast가 진행될수록 많은 business problem에 대응하며 solution을 제공할 수 있을 겁니다. 틀을 만들어 놓으면 간단한 전처리와 hyperparameter tuning만으로 공수 소진을 하는 등의 방법을 고안할 수 있겠습니다.

1. Deploy
2. Dashboard

로 구성돼 있습니다.
첨부한 태블로 파일을 보시며 읽어보시기 바랍니다.
가상환경에서 tabpy를 켜고 여셔야 합니다.

이번에는 PCA를 태블로에서 구현해 어떤 분석을 할 수 있을지 알아보는 시간입니다.
시각화라는 주제에 맞게 꽃길을 걷고 있는 피카츄로 선정했습니다.
자꾸 보고 싶쥬? 아주 귀엽습니다. 얘가 쥐인가요 토끼인가요.

1. Deploy

먼저 파이썬에서 태블로용 PCA 함수를 만들어 deploy 해야겠죠?

몇 가지 오류처리를 한 태블로가 배포한 파이썬 파일도 있지만 너무 복잡해 보입니다.
간단하게 우리가 만들어 보겠습니다.

눈여겨 보셔야할 부분은 역시 input, output 입니다.
input, output의 형식만 알면 어떻게든 처리할 수 있을테니까요.
태블로는 prep을 쓰지 않는 이상 항상 list 로 들어가서 list로 나옵니다.

우선 태블로에서 어떻게 코드가 짜여지는지 보겠습니다.

태블로 파일의 계산된 필드 PCA component 1 입니다.
PCA만 돌아가기 때문에 매우 짧죠? 코드 전체가 다 들어갔습니다.

위 그림에서는 각 _argN 들이 순서대로 list로 들어갑니다.

SUM([City MPG]) 가 _arg1 자리로 들어가, mpg라는 이름의 column으로 dataframe에 저장됩니다.
SUM 으로 나와있지만 'City MPG' column 의 값들 358행이 다 들어가는 겁니다.

SUM([Cyl]) 가 _arg2 자리로 들어가고, Cyl 이라는 이름의 column으로 dataframe에 저장됩니다.

이렇게 만든 dataframe 을 scaling 하고 PCA 돌린 코드가 다음에 나오네요.
PCA score 를 comps 에 저장한 후,
Selected PCA Component 1 매개변수에서 선택한 값을 _arg8로 받고 그 숫자에 해당하는 PCA score를 list로 리턴하도록 만들어져 있습니다.

아래는 또 다른 방식으로 필드를 만든 겁니다.

PCA1_query Cars

계산 결과: 테이블(옆으로) 기준

```
SCRIPT_REAL("return tabpy.query('PCA',_arg1,_arg2,_arg3,_arg4,_arg5,_arg6,_arg7,_arg8,_arg9)['response']",
[PCA1],
SUM([City MPG]),
SUM([Cyl]),
SUM([Dealer Cost]),
SUM([Engine Size]),
SUM([HP]),
SUM([Len]),
SUM([Width]),
SUM([Retail Price]))
```

이건 파이썬에서 함수를 deploy하고 tabpy.query를 이용해 결과를 불러오는 필드입니다.
첫 parameter로 deploy했던 함수 이름을 씁니다.
_arg1을 PCA1 매개변수로 하고, 나머지는 다 column입니다.
뒤에 [‘response’]를 꼭 같은 줄에 붙여주세요. 다음 줄에 쓰면 에러가 나더라고요.

그럼 어떻게 deploy 했는지가 궁금해지시죠?

```
def PCA(_arg1, _arg2, *_argN):
    import numpy as np
    import pandas as pd
    from sklearn.decomposition import PCA
    from sklearn.preprocessing import StandardScaler

    cols = [_arg2] + list(_argN)
    df = pd.DataFrame(data=cols).transpose()

    scale = StandardScaler()
    dat = scale.fit_transform(df)

    pca = PCA()
    comps = pca.fit_transform(dat)  # (358,p)

    return comps[:, _arg1[0] - 1].tolist()
```

deploy 함수입니다.

list를 개수를 모르겠지만 되는대로 받아서,
모듈 import 하고,
dataframe에 넣고 transpose,
scaling,
pca fit,
그리고 PCA score 중 원하는 score를 list로 return 입니다.

역시 input, output이 문제입니다. 들어온 걸 cols에 저장하고 dataframe으로 만드는데요.
데이터가 어떻게 들어오는지 감이 잘 안오니 list를 만들어서 연습합시다.
손맛이 없으면 머리에도 안 들어오쥬?

```
a1 = [1] * 100
a2 = [2] * 100
a3 = [3] * 100
[a1]
list([a2, a3])
cols = [a1]+list([a2, a3])
cols
len([a1]+list([a2, a3]))


import pandas as pd
pd.DataFrame({'m': a1})
pd.DataFrame()
pd.DataFrame(data=cols)
pd.DataFrame(data=cols).transpose()
```

input이 어떻게 돌아가나 궁금하실 분들을 위해 예시를 만들었습니다.
대문짝만하네요. 여러분의 눈을 위해 줄이지 않았습니다.
차례대로 실행하시다보면 어떻게 돌아간다는 건지 이해하실 수 있겠습니다.

```
from tabpy.tabpy_tools.client import Client

client = Client('http://localhost:9004/')
client.deploy('PCA', PCA, 'This is using sklearn PCA',override=True)
```

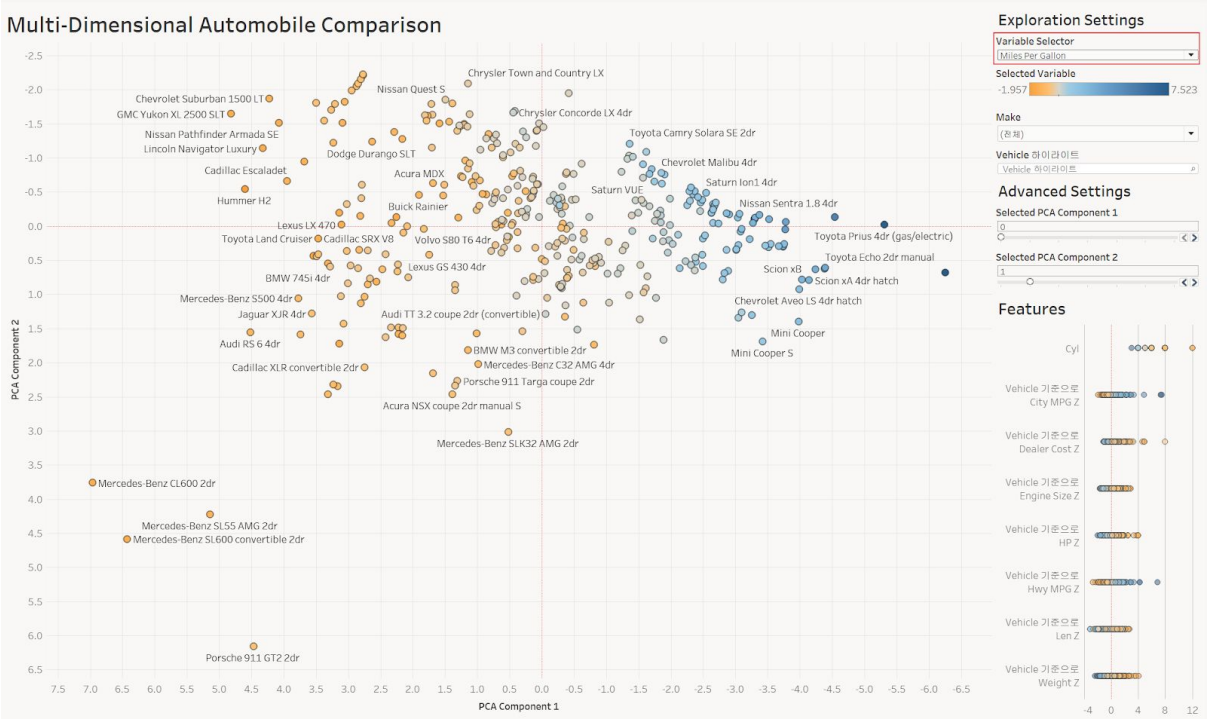
마지막으로 deploy 할 차례입니다.
형식은 이미 알아봤으니 문제없쥬?
에러가 나시나요? tabpy는 키셨나요? 가상환경에서?

2. Dashboard

이제 대시보드를 한 번 살펴보겠습니다.
해당 대시보드는 Cars.csv 의 변수 중 일부분을 PCA 한 결과입니다.

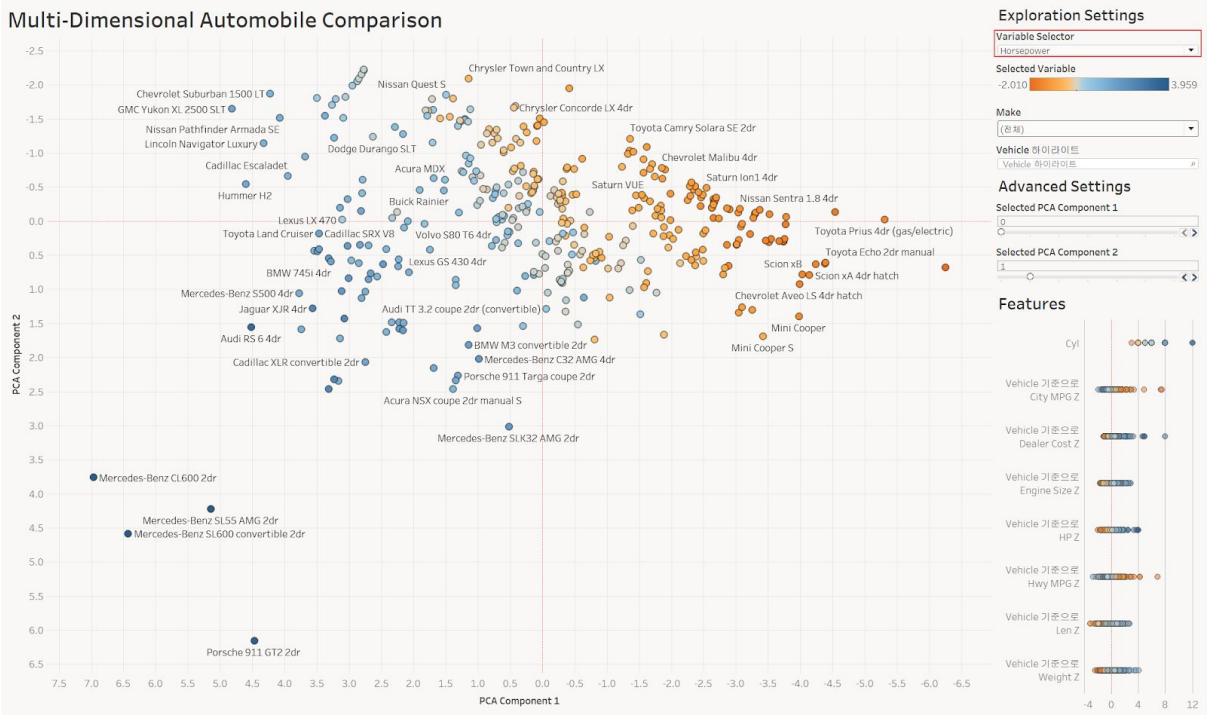
위에서 본 필드와 매개변수,
그리고 평균 0, 분산 1로 만들어 나타낸 분포가 오른쪽 하단에 나타나 있습니다.

다들 태블로 장인이시니, 공유해드린 태블로 파일을 열고 어떻게 생겼나 전체적으로 직접 한 번 보신 후 같이 보시쥬.

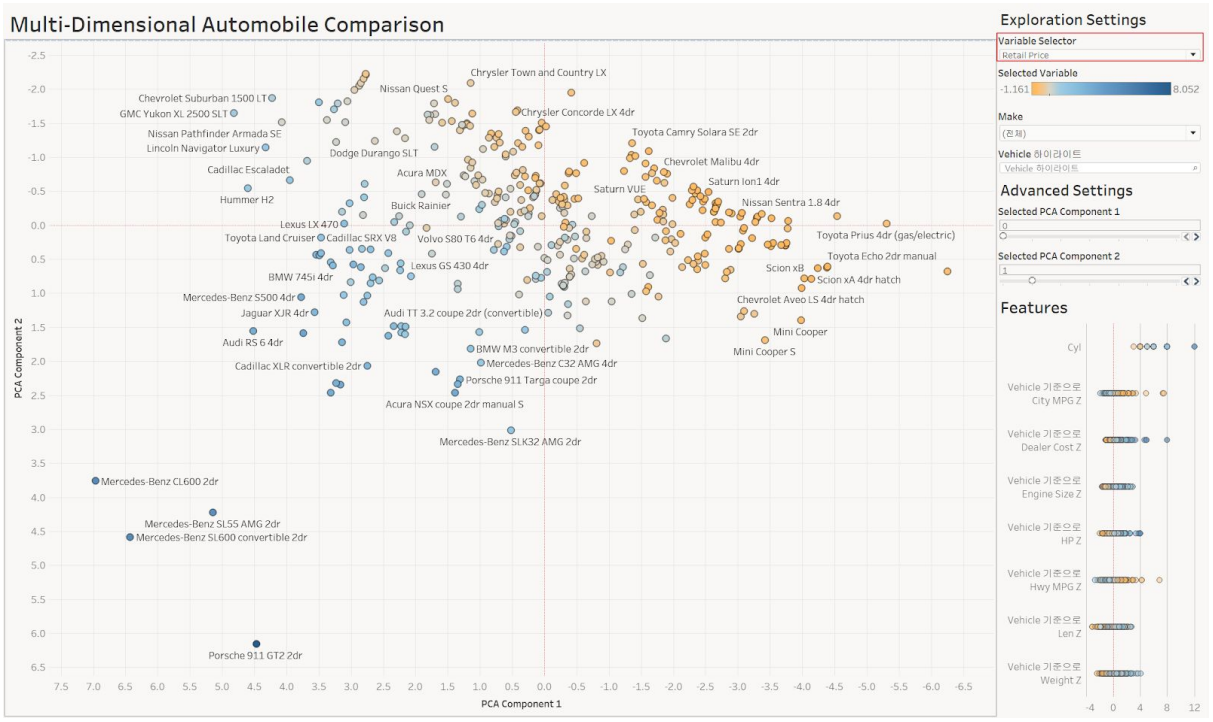


오른쪽 상단에 Variable Selector가 있습니다.
그 중 MPG(Mile Per Gallon)을 선택하시면 위와 같은 화면이 나옵니다.

오른쪽이 연비가 높고 왼쪽으로 갈수록 낮네요.
차종을 보시면 Mercedes나 Porsche의 특정 기종이 연비가 가장 낮고,
Toyota는 연비가 매우 좋은 차종을 공급하고 있네요.

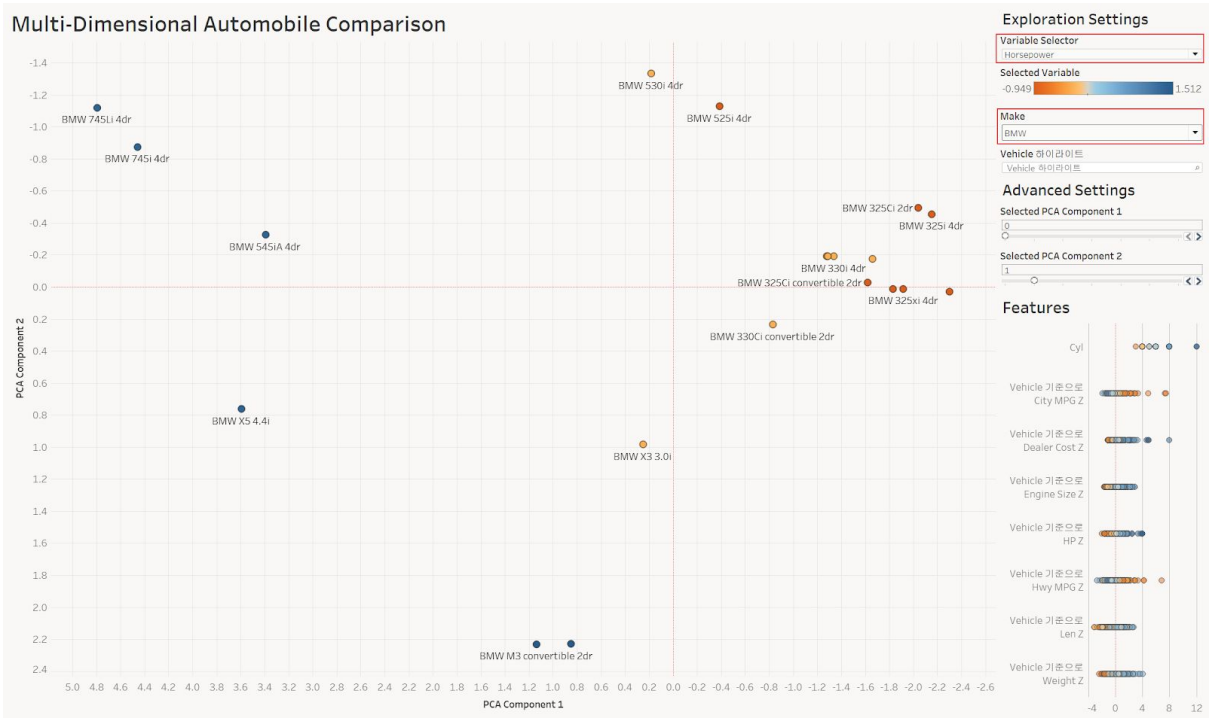


이번엔 마력. Horse Power로 바꿔보겠습니다.
위와 정반대가 되네요.
연비가 안 좋은 대신 폭발력이 있나봅니다.

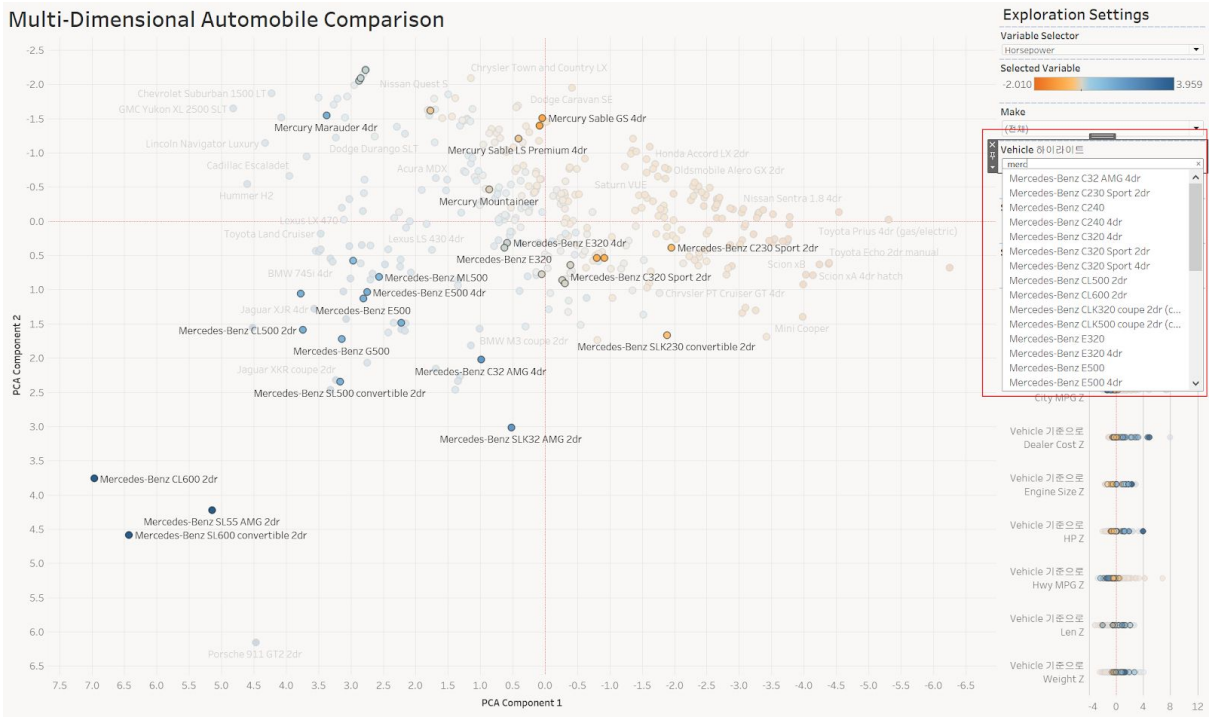


이번엔 Retail Price입니다.
가격도 마력과 비슷한데요.
다만 색깔로 봤을 때 높은 것과 낮은 것의 차이 비율이 마력보다는 적네요.

feature마다 바꿔서 다 보시면 어느 정도 규칙성을 발견할 수 있습니다.
왼쪽 아래, 그러니까 PC1, PC2 score가 높은 곳이
화려하고 스펙이 좋은, 비싸지만 연비가 좋지 않은 차종이고요.
오른쪽으로 갈수록 수수하고 저렴하지만 연비가 좋은 차종입니다.

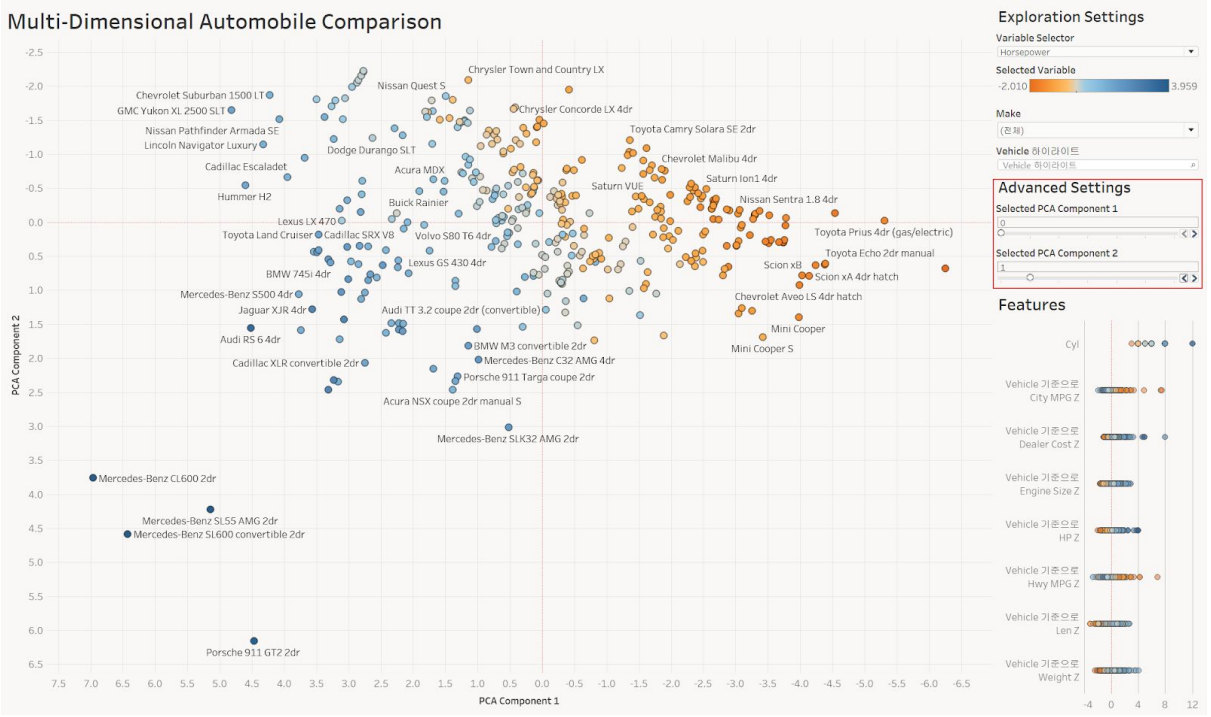


Make를 눌러보시면 제조회사를 선택할 수 있습니다.
각 제조회사가 어떤 차를 만들었는지 볼 수 있네요.
variable selector도 바꿔보시기 바랍니다.
생각보다 다들 골고루 만들었습니다.
제조회사별 분석도 할 수 있겠네요.
제가 차에 관심이 없어서 더 많은 분석을 하지 못해 아쉽습니다.



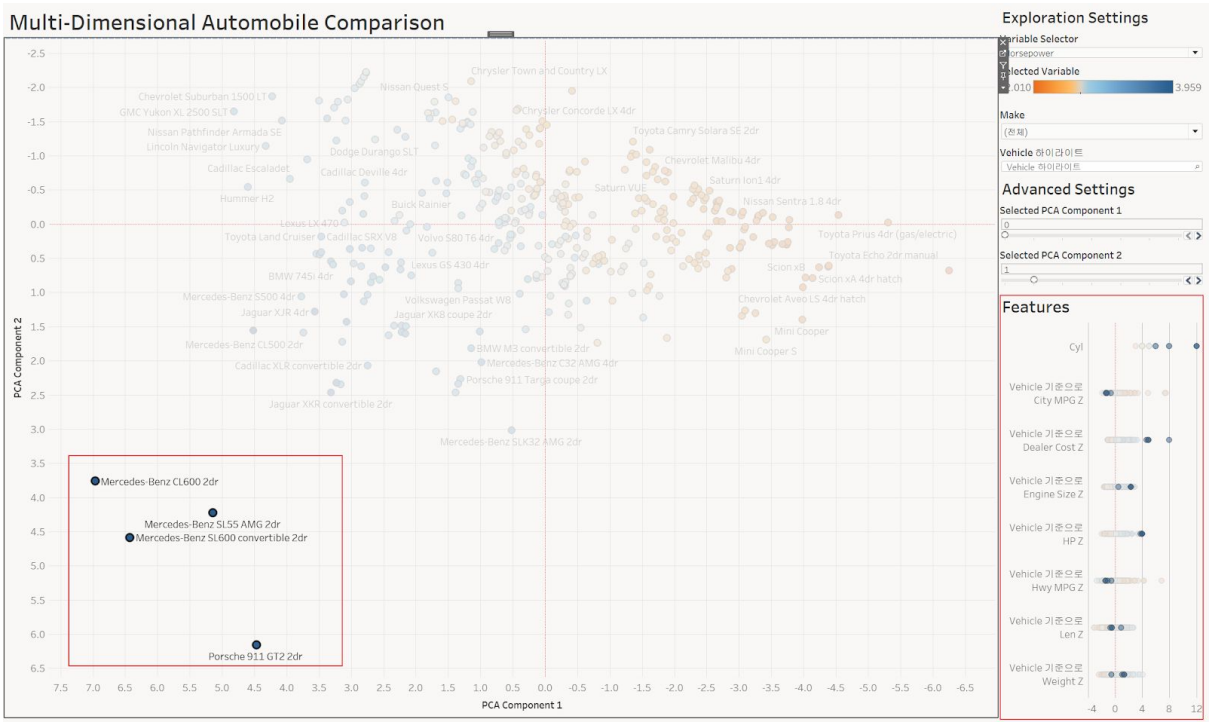
이번엔 차종을 검색할 수 있습니다.

vehicle에서 merc 를 쳐봤더니 Mercedes Benz 사의 차들만 highlight 되네요.
해당 차종들이 어떤 분포를 갖고 있는지 볼 수 있습니다.
얘들은 보통 연비가 안 좋아도 퍼포먼스에 중점을 뒀나봅니다.

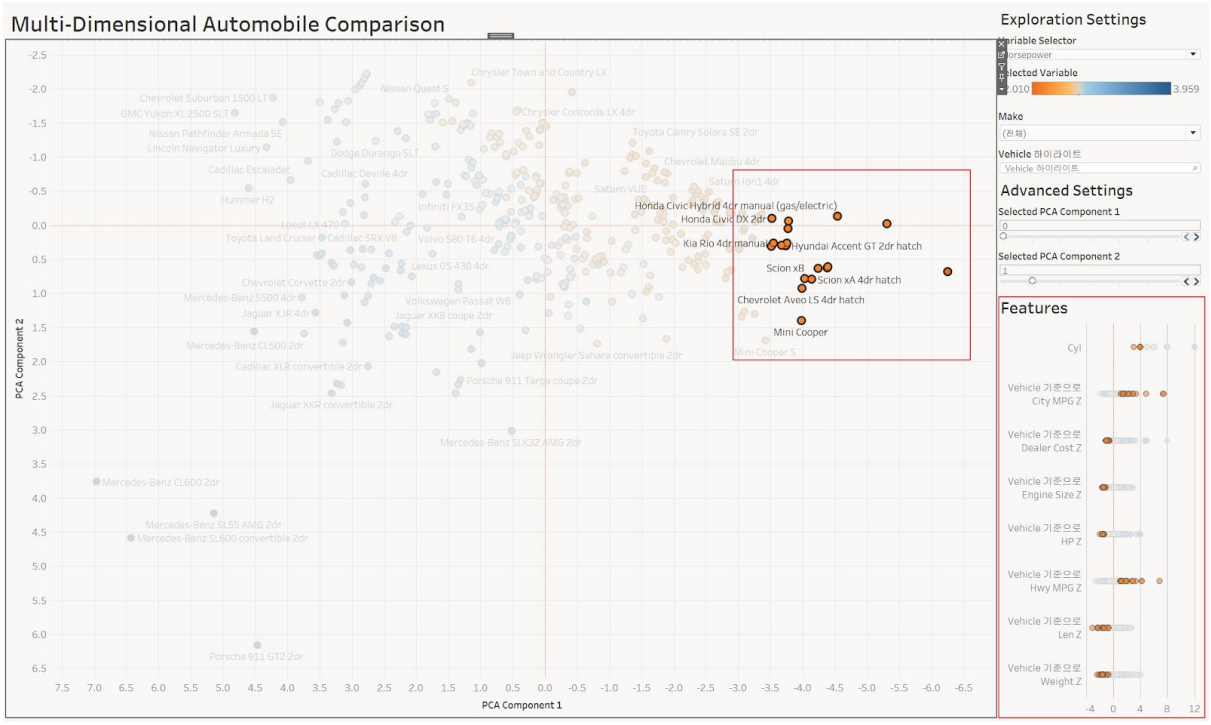


Advanced Settings 에서는 가로축, 세로축에 해당하는 주성분을 바꿀 수 있습니다.
파이썬은 0부터 시작해서 숫자가 다르게 느껴지실 겁니다.
계산된 필드의 return 값을 보시면, 이 매개변수에 해당하는 column 을 가져오게 돼 있습니다.
그래서 0을 선택하면 comps[:, 0] 이 return 됩니다.
그러니까 score의 첫 번째 column 인거죠.
그게 PC1 score입니다.
1을 선택하면 comps[:, 1] 이 return되고 PC2 score가 들어오겠죠?

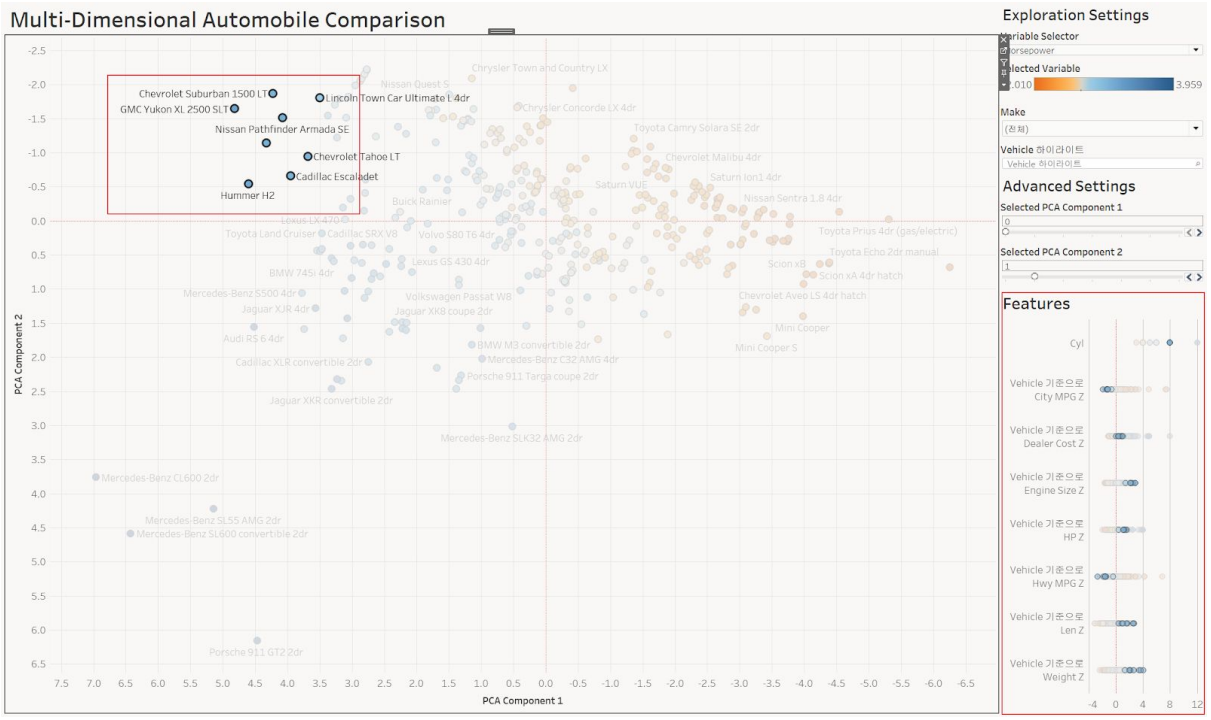
값을 바꿔보면서 scatter plot 이 어떤 모양이 되는지 보시기 바랍니다.
또 모양에 따라 variable도 바꿔보셔서 스토리를 만들어낼 수 있습니다.



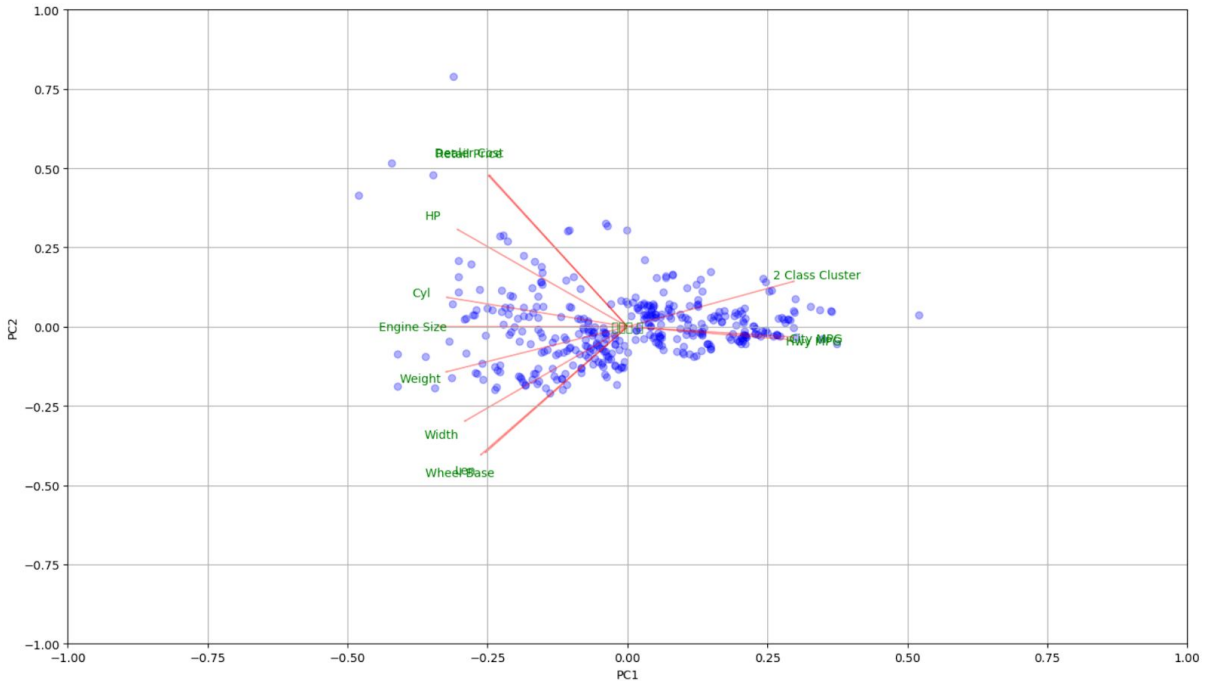
이번엔 부분적으로 선택해보겠습니다.
왼쪽 아래, 그러니까 PC1, PC2 score가 높은 곳을 선택합니다.
그리고 오른쪽 하단 컬럼별 평균0, 분산1이 된 분포를 같이 보겠습니다.
해당 네 차종이 컬럼별로 분포한 위치를 보실 수 있네요.
역시 연비가 낮고 가격이 높은 걸 볼 수 있습니다. 엔진 사이즈도 크고 마력도 높네요.



이번엔 반대편 오른쪽을 선택해보겠습니다.
낮은 퍼포먼스, 높은 연비, 작고 가볍고 저렴한 차종들이네요.



이번엔 왼쪽 상단입니다.
PC1 score는 높지만 PC2 score가 낮습니다.
연비가 낮고 차체가 크고 무겁습니다.
마력은 중간정도지만 가격이 비싸군요.



위 그림처럼 loading vector가 그려지면 좋겠지만 아직 태블로에서 해당 정보를 나타내기 어려운 문제가 있습니다.
어쨌든 지금 그림은 전체 column을 다 PCA 했을 때 결과라 위 대시보드와는 조금 다릅니다.

하지만 연비가 나머지 성능과 반대가 되고 column 별 특성 방향을 알 수 있다는 것에 집중해주세요.
위 biplot 과 같이 대시보드를 해석하시면 좋겠습니다.

감사합니다.