

포드 v 페라리

랩타임 예측 기반 자동차 벤치마킹 모델



7000RPM 김정환 임요셉 정지혁 조재훈

목차

Lap 1 / 프로젝트 소개

Lap 2 / 회귀 분석

Lap 3 / 결론 및 의의

Lap 4 / 한계점



LE MANS '66

Lap
1/4



최신 양산차들의 성능 시험 무대 / 랩 타임 기록



Rank	Vehicle	Driver	Time	PS / KG
1.	Porsche 911 GT2 RS Manthey Racing	Lars Kern	6:40.33	- / -
2.	AMG GT Black Series	Maro Engel	6:43.62	730 / 1633
3.	Lamborghini Aventador SVJ	Marco Mapelli	6:44.97	770 / 1718
4.	Radical SR8LM	Michael Vergers	6:48.00	455 / 650
5.	Huracán Performante	Marco Mapelli	6:52.01	640 / 1556
6.	Radical SR8	Michael Vergers	6:55.00	363 / 650
7.	Porsche 911 GT3 RS	Kevin Estre	6:56.40	520 / 1444
8.	Porsche 918 Spyder	Marc Lieb	6:57.00	887 / 1654
9.	Modified Subaru WRX STI	unknown	6:57.50	- / -
10.	Aventador LP750-4 SV	unknown	6:59.73	750 / 1699
11.	Ferrari 488 Pista	Christian Gebhardt	7:00.03	720 / 1476
12.	Dodge Viper ACR (Mk V)	Lance Arnold	7:01.30	654 / 1536
13.	Mercedes-Benz AMG GT R Pro	Maro Engel	7:04.63	585 / 1640
14.	Nio EP9	unknown	7:05.12	1360 / 1735
15.	Mercedes-Benz AMG GT R Pro	Christian Gebhardt	7:07.00	585 / 1640
16.	McLaren 720S	Christian Gebhardt	7:08.00	720 / 1429
17.	Nissan GT-R Nismo (R35)	Michael Krumm	7:08.68	600 / 1745
18.	McLaren 600LT	Christian Gebhardt	7:08.82	600 / 1404
19.	Mercedes-Benz AMG GT R	Christian Gebhardt	7:10.92	585 / 1638
20.	Gumpert Apollo Sport	Florian Gruber	7:11.57	700 / 1200
21.	Dodge Viper SRT-10 ACR	Dominik Farnbacher	7:12.13	608 / 1536
22.	Porsche 911 GT3	unknown	7:12.70	500 / 1469

H 한국경제

현대차, '뉘르부르크링 24시 내구레이스' 5년 연속 완주
현대차는 26일부터 27일(현지시간)까지 독일 뉘르부르크링 서킷에서 열린 '뉘르부르크링 24시 내구레이스'에 고성능 차량 '벨로스터 N TCR', 'i30 ...
2020. 9. 28.



오토헤럴드

현대차 신형 투싼, 뉘르부르크링에서 알프스까지 섭렵한 주행성능
현대차는 최근 몇 개월 동안 신형 투싼은 세계에서 가장 까다로운 트랙으로 알려진 뉘르부르크링 노르트슬라이페 서킷에서 고속 주행과 내구성 ...
2020. 9. 9.



모터그래프

포르쉐 타이칸, 뉘르부르크링 랩 타임 결과는?
포르쉐 순수전기스포츠카 타이칸이 독일 뉘르부르크링 노르트슬라이페 서킷에서 7분 42초의 랩 타임 기록을 세웠다.
2019. 8. 28.



모터그래프

테슬라, 차세대 '로드스터' 뉘르부르크링 테스트 예정
테슬라 일론 머스크 최고 경영자에 따르면, 내년 독일 뉘르부르크링 서킷에서 차세대 로드스터의 랩 타임 테스트를 진행할 예정이다. 지난달 포르쉐가 ...
2019. 9. 15.



“어떤 부품을 튜닝해야
기록 단축에 효과적일까?”



“회귀 분석을 통해 알아보자 ”

변수 설정 및 데이터 크롤링

```

1 url_list = dic["url"]
2 spec_dic = {"CurbWeight": [], "TopSpeed": [], "Displacement": [], "Power": [], "Torque": [], "Zero": [] }
3 for url in url_list:
4     if url == [] :
5         # URL이 없을 경우 결측치로 입력
6         spec_dic["CurbWeight"].append("")
7         spec_dic["TopSpeed"].append("")
8         spec_dic["Displacement"].append("")
9         spec_dic["Power"].append("")
10        spec_dic["Torque"].append("")
11        spec_dic["Zero"].append("")
12        continue
13
14 res = rq.get(addr+url)
15 my_html = res.text
16 my_soup = bs4.BeautifulSoup(my_html, 'html.parser')
17 spec = my_soup.select("tr > td")
18 Power, TopSpeed, Torque, CurbWeight, Displacement, Zero = ...
19 for i in range(len(spec)):
20     if (spec[i].text == "Power"):
21         Power = re.findall("#d+.#d+", spec[i+1].text)[0]
22     if (spec[i].text == "Top speed"):
23         TopSpeed = re.findall("#d+.#d+", spec[i+1].text)[0]
24     if (spec[i].text == "Torque"):
25         Torque = re.findall("#d+.#d+", spec[i+1].text)[0]
26     if (spec[i].text == "Curb weight"):
27         CurbWeight = re.findall("#d+", spec[i+1].text)[0]
28     if (spec[i].text == "Displacement"):
29         Displacement = re.findall("#d+.#d+", spec[i+1].text)[0]
30     if (spec[i].text == "0 - 100 kph"):
31         Zero = re.findall("#d+.#d+", spec[i+1].text)[0]
32 spec_dic["CurbWeight"].append(CurbWeight)
33 spec_dic["TopSpeed"].append(TopSpeed)
34 spec_dic["Displacement"].append(Displacement)
35 spec_dic["Power"].append(Power)
36 spec_dic["Torque"].append(Torque)
37 spec_dic["Zero"].append(Zero)

```

```

1 df = pd.DataFrame(spec_dic)
# 크롤링한 데이터를 DataFrame으로 만들어준다.

```

```


1 df["vehicle"] = dic["vehicle"]
2 df["laptime"] = dic["laptime"]
# 앞에서 크롤링한 Laptime, 차종을 합쳐준다.

```

```

1 df.to_excel("vehicle_zero.xlsx")
# 크롤링한 데이터를 Excel파일로 만들어준다.

```



Mercedes-Benz AMG GT Black Series specs

Price in Europe €335,240 - €342,780

Car type Coupe

Curb weight 1626-1640 kg (3585-3616 lbs)

Introduced 2020

Origin country Germany

Gas mileage 14.6 l/100 km (16 mpg US / 19 mpg UK)

Views 11.8k

Submitted by benedekpuskas

General performance

Top speed 325 kph (202 mph)

Est. 0 - 100 mph - 9.3 s @ 656 ft 0

Est. max acceleration 0.93 g (9 m/s²)

18m slalom 74.1 kph (46.0 mph)

Est. emissions 349 g/km

Downforce @ 250 kph 400 kg (882 lbs)

100 kph - 0 28 m (92 ft)

200 kph - 0 102 m (335 ft)

Powertrain specs

Engine type V8 Twin Turbo

Displacement 4.0 l (244 ci)

Power 730 ps (720 bhp / 537 kw)

Torque 800 Nm (590 lb-ft)

Power / liter 182 ps (180 hp)

Power / weight 447 ps (441 bhp) / t

Torque / weight 490 Nm (361 lb-ft) / t

Efficiency 50 PS per l/100 km

Power / €5000 11 ps

Layout front engine, rear wheel drive

Lap times

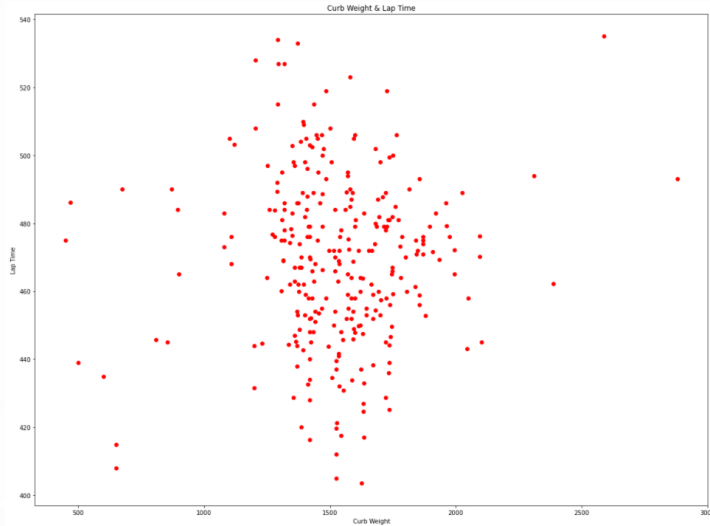
Best ▾ Filters ▾

Track	Time
EuroSpeedway Lausitz	1:25.23
Hockenheim GP	1:43.30
Nürburgring Nordschleife	6:43.62
Nürburgring Nordschleife Full	6:48.05
Sachsenring	1:25.62

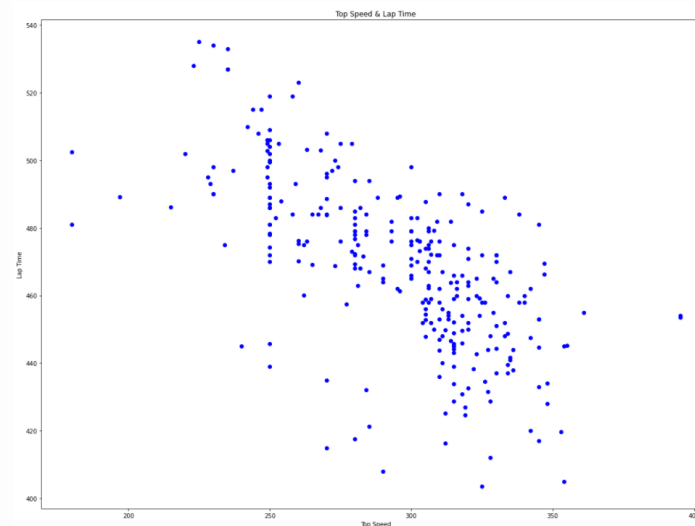
Ads by Google

	A	B	C	D	E	F	G	H
1		CurbWeigh	TopSpeed	isplaceme	Power	Torque	vehicle	laptime
2	1	1626	325	4	730	800	GT Black S	403.62
3	2	1525	354	6.5	770	720	hini Avent	404.97
4	3	650	290	2.8	455	380	adical SR8L	408
5	4	1526	328	5.2	640	600	án Perform	412.01
6	5	650	270	2.6	363	271	Radical SR8	415
7	6	1420	312	4	520	469	he 911 GT	416.4
8	7	1634	345	4.6	887	1280	he 918 Sp	417
9	8	1545	280	2.5	309	393	d Subaru V	417.5
10	9	1525	353	6.5	750	690	ador LP750	419.73
11	10	1385	342	3.9	720	770	rari 488 Pi	420.03
12	11	1527	285	8.4	654	813	Viper ACR	421.3
13	12	1632	319	4	585	700	-Benz AMC	424.63
14	13	1735	312	3.5	1360	1480	Nio EP9	425.12
15	14	1632	319	4	585	700	Benz AMG	427
16	15	1419	348	4	720	770	clLaren 720	428
17	16	1720	315	3.8	600	652	GT-R Nism	428.68
18	17	1356	328	3.8	600	620	clLaren 600	428.82
19	18	1554	318	4	585	700	es-Benz AM	430.92
20	19	1200	327	4.2	700	875	iert Apollo	431.57
21	20	1536	284	8.4	608	759	Viper SRT-	432.13
22	21	1413	320	4	500	460	sche 911 C	432.7
23	22	1634	345	4.6	887	1280	he 918 Sp	433
24	23	1587	315	6.2	659	881	t Corvette	433.9
25	24	1419	348	4	720	770	clLaren 720	434
26	25	1509	326	4.8	570	480	Nurburgrin	434.64
27	26	600	270	1.8	350	500	kervoort D	434.89
28	27	1733	310	6.2	659	881	o ZL1 1LE F	436.04
29	28	1622	330	3.7	650	800	911 Turbo	437
30	29	1525	334	3.8	580	750	he 911 Tu	437.11
31	30	1370	336	3.6	620	700	911 GT2 f	438
32	31	1670	322	5	600	700	r XE SV Pro	438.36
33	32	500	250	1.5	320	289	ical SR3 Tu	439
34	33	1736	315	3.8	550	632	GT-R (R35	439.1
35	34	1525	324	6.2	647	840	Lap 2/4	439.62

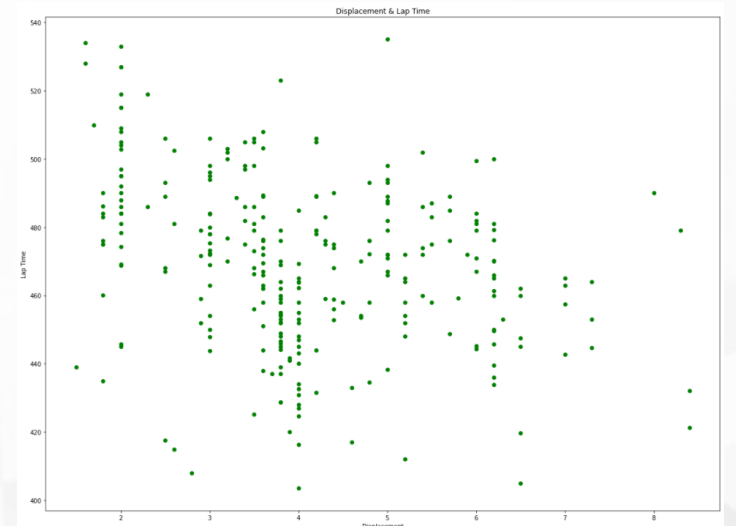
랩타임과 개별 변수간의 시각화



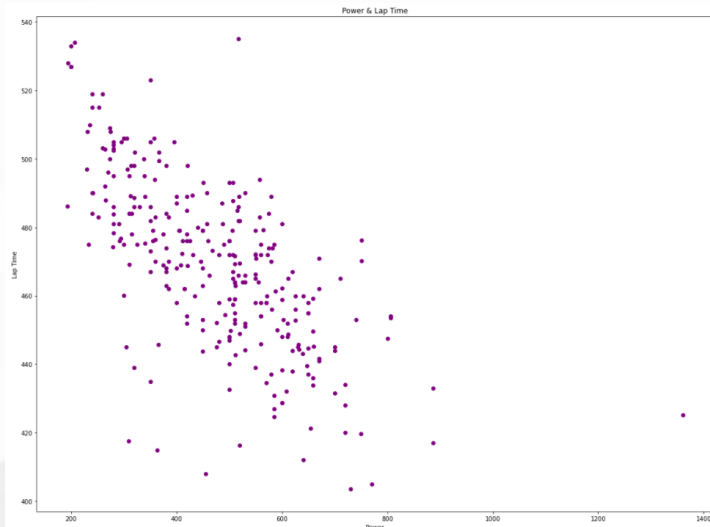
Lap Time VS 차량 중량



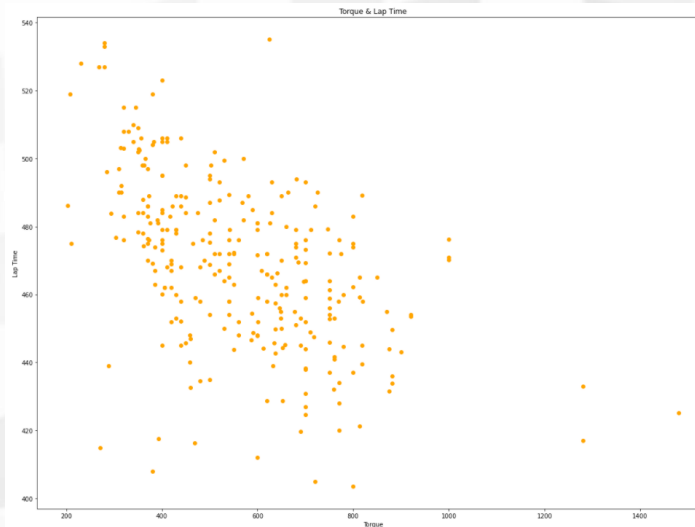
Lap Time VS 최고 속도



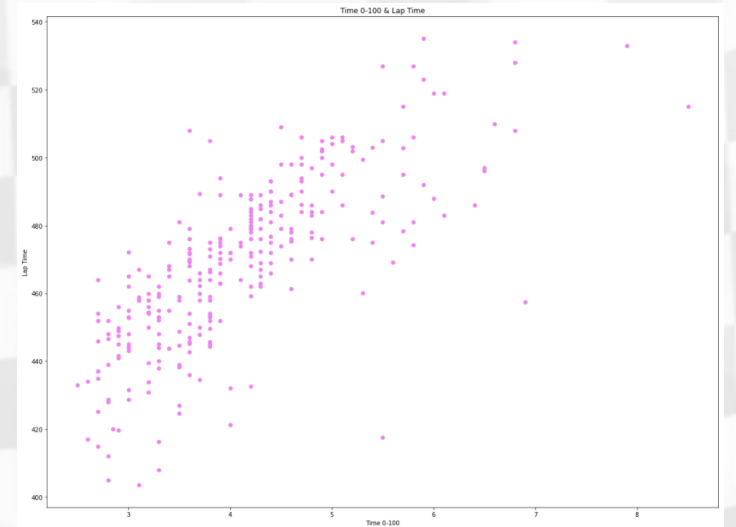
Lap Time VS 배기량



Lap Time VS 마력



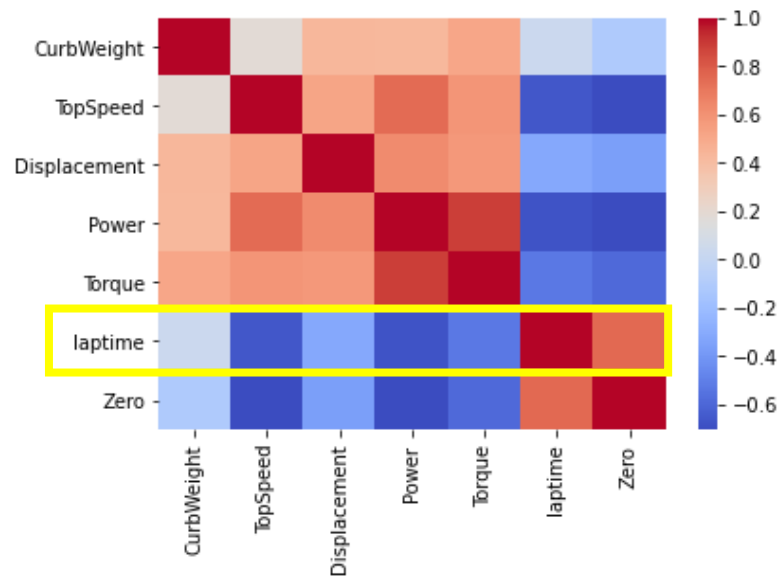
Lap Time VS 토크



Lap Time VS 제로백

회귀 분석 P1: 상관계수 행렬, Statsmodel (OLS)

```
sns.heatmap(df.corr(), cmap='coolwarm')
plt.show()
```



상관계수 행렬 Heatmap

```
myModel1 = smf.ols(formula = "laptime ~ CurbWeight + Displacement + TopSpeed + Power + Torque + Zero", data=df)
result1 = myModel1.fit()
result1.summary()
```

OLS Regression Results

Dep. Variable:	laptime	R-squared:	0.664			
Model:	OLS	Adj. R-squared:	0.657			
Method:	Least Squares	F-statistic:	92.52			
Date:	Wed, 03 Feb 2021	Prob (F-statistic):	1.30e-63			
Time:	10:36:31	Log-Likelihood:	-249.93			
No. Observations:	288	AIC:	513.9			
Df Residuals:	281	BIC:	539.5			
Df Model:	6					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	6.7546	0.426	15.850	0.000	5.916	7.593
CurbWeight	7.643e-05	0.000	0.482	0.630	0.000	0.000
Displacement	0.0495	0.032	1.551	0.122	-0.013	0.112
TopSpeed	-0.0071	0.002	-4.375	0.000	-0.010	-0.004
Power	-0.0015	0.001	-2.100	0.037	-0.003	-9.12e-05
Torque	-0.0004	0.000	-0.805	0.421	0.001	0.001
Zero	0.0002	2.23e-05	7.616	0.000	0.000	0.000
Omnibus:	86.363	Durbin-Watson:	1.778			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	304.838			
Skew:	1.258	Prob(JB):	6.39e-67			
Kurtosis:	7.367	Cond. No.	3.08e+04			

p 값류 검정
(0.05이상)

Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 3.08e+04. This might indicate that there are strong multicollinearity or other numerical problems.

회귀 분석 P2: 선형회귀 모델

```
1 from sklearn.linear_model import LinearRegression
2 from sklearn import metrics, preprocessing
3
```

```
1 lm = LinearRegression(fit_intercept = True) # 선형회귀함수 정의
```

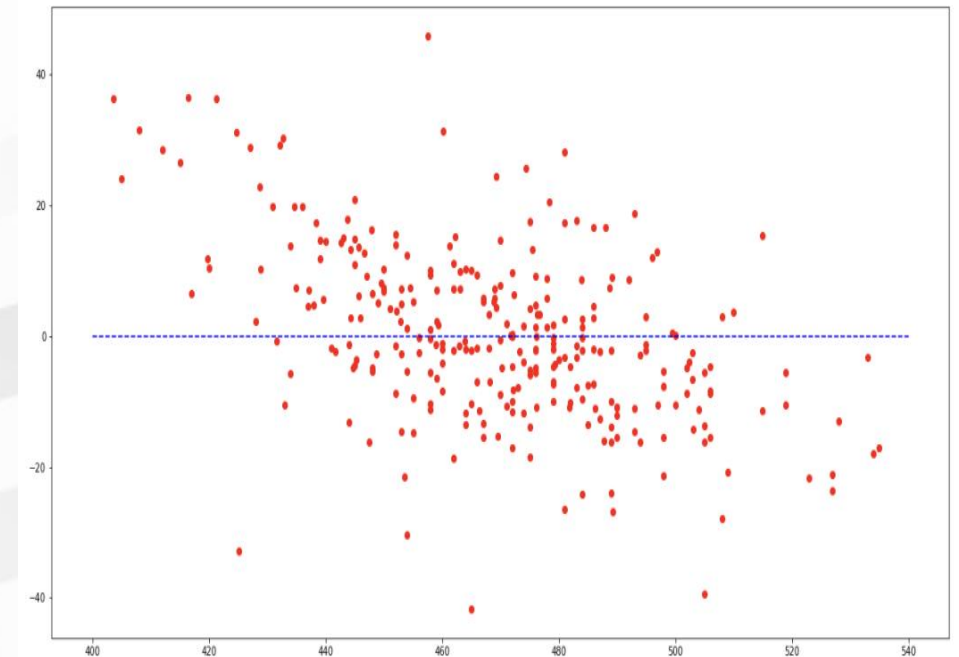
```
1 Y_data = df["laptime"].values # X 데이터, Y 데이터로 나눈 뒤 Train 데이터, Test 데이터로 나눈
2 X_data = df[["CurbWeight", "Displacement", "TopSpeed", "Power", "Torque", "Zero"]].values
3 X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_data, test_size=0.2, random_state=1234)
```

```
1 lm.fit(X_train, Y_train) # Train 데이터로 선형회귀 훈련
```

```
1 Y_pred = lm.predict(X_test) # Test 데이터로 선형회귀 예측 후 RMSE 값 측정
2 difference = lm.predict(X_data) - Y_data
3 print("Tree best RMSE : " + str(np.round(np.sqrt(metrics.mean_squared_error(Y_test, Y_pred)), 3)))
```

```
1 plt.figure(figsize = (20, 10)) # 전체 데이터와 예측값의 차이를 Plot에 시각화
2 plt.plot([400, 540], [0, 0], '--', c = 'blue')
3 plt.scatter(Y_data, difference, c = 'red')
4 plt.show()
```

Tree best RMSE : 12.023



회귀 분석 P2: RandomForestRegressor (GridSearch로 최적화)

```
1 from sklearn.model_selection import train_test_split, GridSearchCV
2 from sklearn.ensemble import RandomForestRegressor
3 from sklearn import metrics
4 from sklearn.datasets import load_digits
```

```
1 depth_grid = np.arange(1, 21) # 하이퍼파라미터의 범위 설정
2 n_estimators = np.arange(50, 200)
3 min_samples_leaf_grid = np.arange(2, 31, 2)
4 max_leaf_nodes_grid = np.arange(2, 51, 2)
5 parameters = {'max_depth': depth_grid, 'min_samples_leaf': min_samples_leaf_grid, 'max_leaf_nodes': max_leaf_nodes_grid}
```

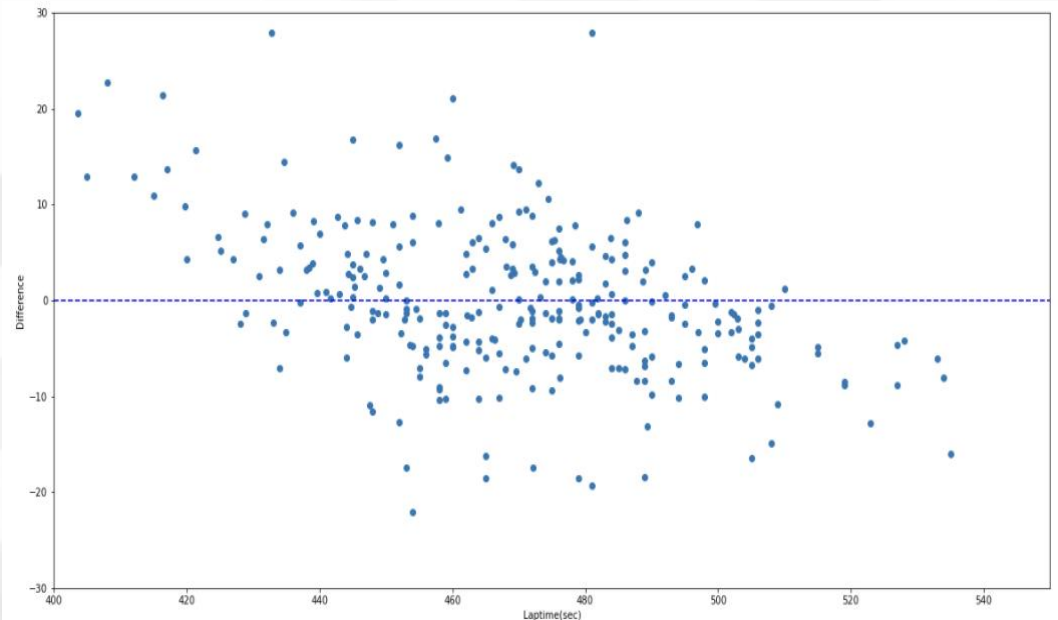
```
1 gridCV = GridSearchCV(RandomForestRegressor(), parameters, cv=10, n_jobs=-1) # RandomForestRegressor 함수 입력
2 gridCV.fit(X_train, Y_train) # 데이터의 최적화된 하이퍼파라미터의 찾기
3 best_depth = gridCV.best_params_['max_depth']
4 best_min_samples_leaf = gridCV.best_params_['min_samples_leaf']
5 best_max_leaf_nodes = gridCV.best_params_['max_leaf_nodes']
```

```
1 RFR_best = RandomForestRegressor(max_depth=best_depth, min_samples_leaf=best_min_samples_leaf, max_leaf_nodes=best_max_leaf_nodes)
2 RFR_best.fit(X_train, Y_train) # 최적화된 하이퍼파라미터를 입력하고 훈련
3 Y_pred = RFR_best.predict(X_test) # 예측모델의 Test 데이터를 이용하여 RMSE 출력
4 print("Tree best RMSE : " + str(np.round(np.sqrt(metrics.mean_squared_error(Y_test, Y_pred)), 3)))
5 Y_pred_RFR = RFR_best.predict(X_data_)
```

```
1 plt.figure(figsize=(20, 10)) # 전체 데이터와 예측값의 차이를 Plot에 시각화
2 plt.plot([400, 540], [0, 0], '--', c='blue')
3 plt.xlabel("Laptime(sec)")
4 plt.ylabel("Difference")
5 plt.scatter(Y_data, Y_pred_RFR - Y_data)
6 plt.show()
```

Tree best SCORE : 0.69

Tree best RMSE : 11.518



회귀 분석 P2: XGBoostRegressor (GridSearch로 최적화)

```
1 Y_data_ = df["laptime"].values
2 # X_data_ = df[["CurbWeight", "TopSpeed", "Power", "Torque", "Zero"]].values
3 X_data_ = df[["CurbWeight", "Displacement", "TopSpeed", "Power", "Torque", "Zero"]].values
4 X_train, X_test, Y_train, Y_test = train_test_split(X_data_, Y_data_, test_size=0.2, random_state=1234)
```

```
1 depth_grid = np.arange(1,21) # 하이퍼파라미터의 범위 설정
2 n_estimators=np.arange(50,200)
3 min_samples_leaf_grid = np.arange(2,31,2)
4 max_leaf_nodes_grid = np.arange(2,51,2)
5 parameters = {'max_depth':depth_grid, 'min_samples_leaf':min_samples_leaf_grid, 'max_leaf_nodes':max_leaf_nodes_grid }
```

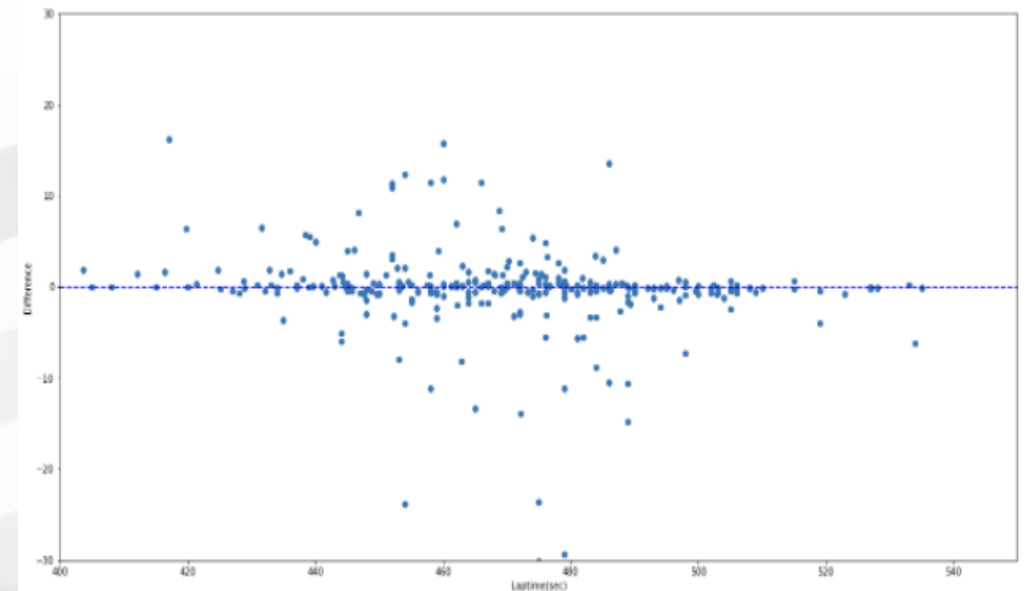
```
1 gridCV = GridSearchCV(XGBRegressor(), parameters, cv=10, n_jobs = -1) # XGBRegressor 함수 입력
2 gridCV.fit(X_train, Y_train) # 데이터의 최적화된 하이퍼파라미터의 찾기
3 best_depth = gridCV.best_params_['max_depth']
4 best_min_samples_leaf = gridCV.best_params_['min_samples_leaf']
5 best_max_leaf_nodes = gridCV.best_params_['max_leaf_nodes']
```

```
1 XGBR_best = XGBRegressor(max_depth=best_depth,min_samples_leaf=best_min_samples_leaf,max_leaf_nodes=best_max_leaf_nodes)
2 XGBR_best.fit(X_train, Y_train) # 최적화된 하이퍼파라미터를 입력하고 훈련
3 Y_pred = XGBR_best.predict(X_test) # 예측모델의 Test 데이터를 이용하여 RMSE 출력
4 print( "Tree best RMSE : " + str(np.round(np.sqrt(metrics.mean_squared_error(Y_test,Y_pred)),3)))
```

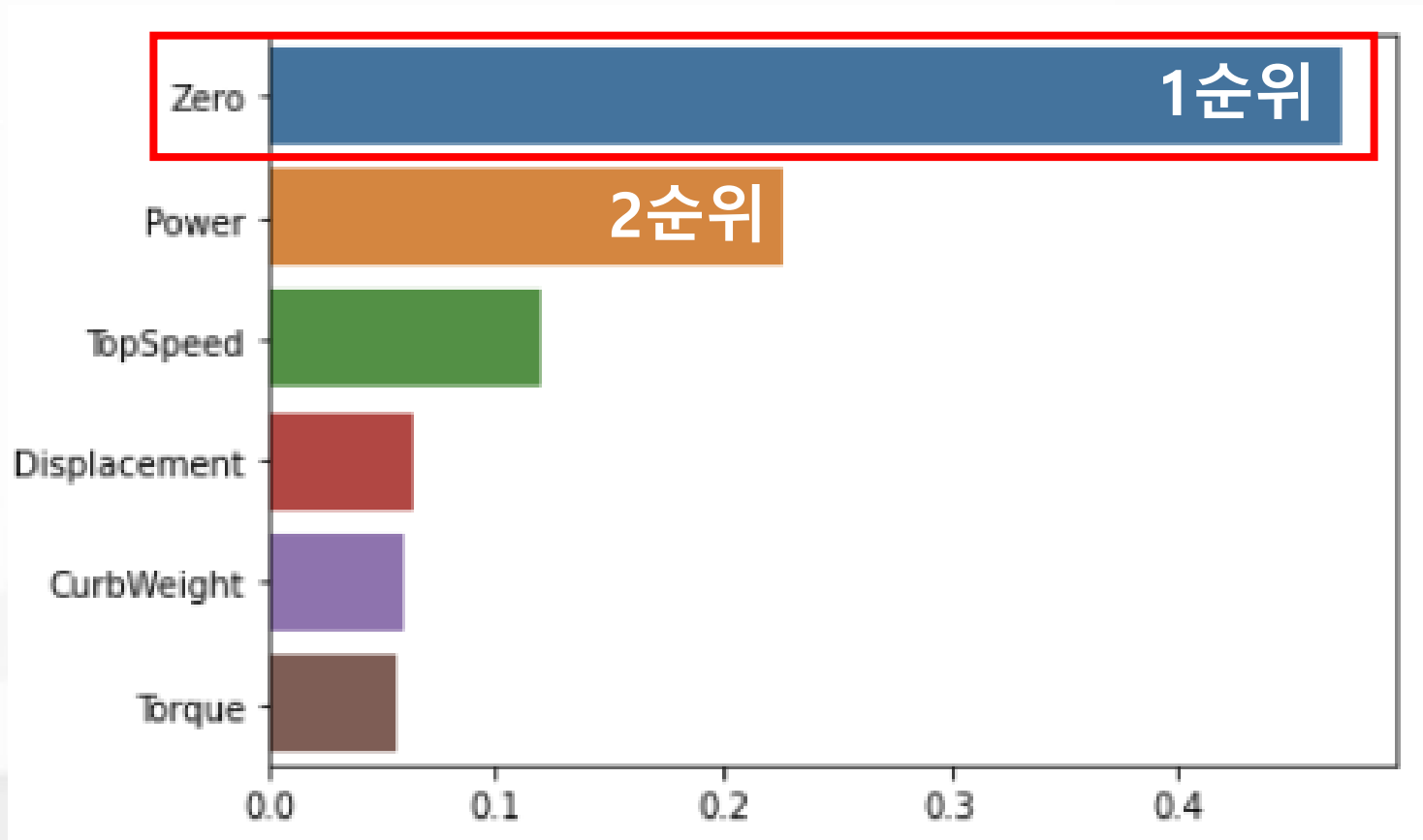
```
1 plt.figure(figsize = (20,10)) # 예측모델의 Test 데이터를 이용하여 RMSE 출력
2 plt.plot([400,540],[0,0], '--', c='blue')
3 plt.xlabel("Laptime(sec)")
4 plt.ylabel("Difference")
5 plt.scatter(Y_data_,Y_pred,XGBR-Y_data_)
6 plt.show()
```

Tree best SCORE : 0.7

Tree best RMSE : 11.416



회귀 분석 P2: XGBoostRegressor (Feature Importance로 추출)



결론 및 의의

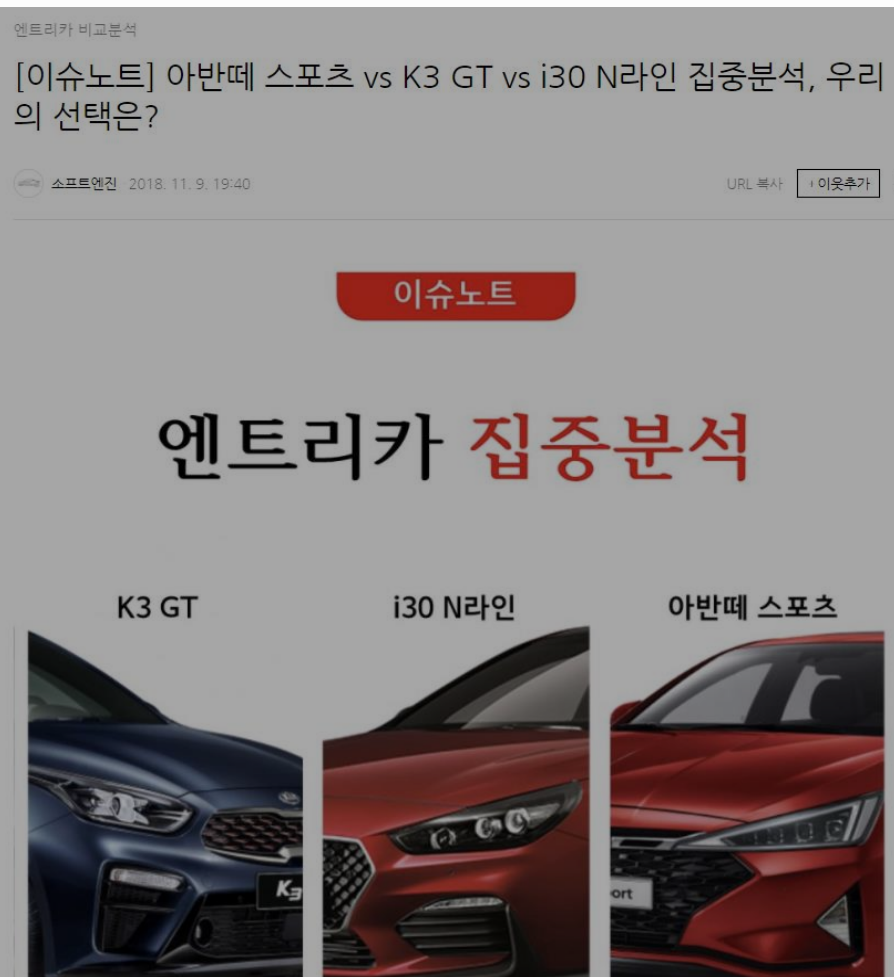
회귀분석 1: 튜닝할 때 중량과 토크 값에는 신경을 쓰지 않아도 된다

회귀분석 2: 따라서 랩타임에 가장 영향을 미치는 요소는 제로백이므로 제로백을 낮추는 것에 포커스를 맞추어 튜닝을 한다.

- 모든 변수들이 랩 타임에 중요한 것은 사실, 하지만 우선순위 결정에 도움을 준다는 점에 의의
- 그 밖에 2순위인 최고속력 등의 변수들도 조합하여 레이싱 테스트 비용 절감에 도움을 줄 수 있음

따라서 벤치마킹 툴로써 유효

추가 활용 방안: 그래서 포드(i30) vs 페라리(K3 GT) 누구 이길까?



차종	Laptime
G70 3.3 터보가솔린	: 485.95
스팅어 3.3 터보가솔린	: 491.63
K9 3.3 터보가솔린	: 502.52
G90 3.3 터보가솔린	: 507.32
그랜저 3.3 가솔린	: 517.7
말리부 2.0터보	: 519.73
그랜저 2.4 가솔린	: 528.05
K7 2.5 가솔린	: 528.05
팔리세이드 2.2디젤	: 523.0
i30N 라인	: 519.46
K3 GT(5도어)	: 527.8
G80 3.3 가솔린	: 503.33
코나 1.6 터보가솔린	: 526.63
셀토스 1.6터보가솔린	: 526.58
쏘울 EV	: 524.61
니로 EV	: 524.89
스포티지 2.0 디젤	: 516.63
K5 2.0 가솔린	: 513.31
쏘나타2.0 가솔린	: 512.98
투싼2.0 디젤	: 516.63
티볼리 1.6 디젤	: 522.49
베뉴 1.6 가솔린	: 525.76
K3 1.6 가솔린	: 517.59
아반떼 1.6 가솔린	: 519.07
카니발 2.2디젤	: 516.05
스파크 1.0 가솔린	: 524.97
모닝 1.0 가솔린	: 525.76
레이 1.0 가솔린	: 525.76
스타렉스 2.5 디젤	: 524.06

“i30 N라인 승!”

한계점

- 주행 성능 이외의 안전, 연비 등의 자동차 전반에 대한 평가 불가
- 기타 성능에 요인을 미치는 변수의 부재 (타이어 상태, 서스펜션 세팅 등)



Finish

감사합니다