به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

هوش مصنوعی

گزارش تمرین کامپیوتری صفر

نام و نام‌خانوادگی: نرگس غلامی

شماره دانشجویی: ۸۱۰۱۹۸۴۴۷

هدف پروژه:

در این پروژه ما می‌خواهیم که نواقص یک دیتا را با پیش‌بینی‌هایی که به کمک تحلیل آماری انجام می‌شود پر کنیم و طرز استفاده از vectorization را یاد بگیریم.

توضیح پروژه:

در این پروژه ما ابتدا یک داده اولیه در مورد یک سری ماشین داریم که با استفاده از این اطلاعات ما قرار است پیش‌بینی کنیم که هر ماشین چقدر کربن دی‌اکسید تولید می‌کند. در ابتدای پروژه با دیتافریم آشنا می‌شویم و انواع توابع مربوط به آن آن را بررسی می‌نماییم، سپس پس از visualize کردن داده‌ها بهترین ویژگی را انتخاب می‌کنیم و با استفاده از یک تقریب خطی نواقص دیتا را پیش‌بینی می‌نماییم.

بخش اول:

```
data = pd.read_csv('FuelConsumptionCo2.csv')
fuelData = pd.DataFrame(data)
print(fuelData.head())
print(fuelData.describe())
print(fuelData.tail())
```

ابتدا فایل FuelConsumptionCo2.csv را باز می‌کنیم. این فایل حاوی دیتای مربوط به ماشین‌ها است. ما با استفاده از این دیتا می‌خواهیم نواقص داده‌های داخل همین دیتا را پیش‌بینی کنیم.

تابع head : این تابع ردیف‌های بالای یک DataFrame یا سری را برمی گرداند که n مقدار ورودی کاربر است. در این بخش ما برای head ورودی تعیین نکردیم در نتیجه فقط ۵ تا از ردیف‌ها را بر می‌گرداند.(مقدار دیفالت آن ۵ ردیف است)

خروجی این بخش:

```
   Unnamed: 0  MODELYEAR   MAKE       MODEL VEHICLECLASS  ENGINESIZE  \
0           0       2014  ACURA         ILX      COMPACT         2.0
1           1       2014  ACURA         ILX      COMPACT         2.4
2           2       2014  ACURA  ILX HYBRID      COMPACT         1.5
3           3       2014  ACURA     MDX 4WD  SUV - SMALL         3.5
4           4       2014  ACURA     RDX AWD  SUV - SMALL         3.5

   CYLINDERS TRANSMISSION FUELTYPE  FUELCONSUMPTION_CITY  FUELCONSUMPTION_HWY  \
0        4.0          AS5        Z                   9.9                  6.7
1        4.0           M6        Z                  11.2                  7.7
2        4.0          AV7        Z                   6.0                  5.8
3        6.0          AS6        Z                  12.7                  9.1
4        6.0          AS6        Z                  12.1                  8.7

   FUELCONSUMPTION_COMB  FUELCONSUMPTION_COMB_MPG  CO2EMISSIONS
0                   8.5                        33         196.0
1                   9.6                        29         221.0
2                   5.9                        48         136.0
3                  11.1                        25           NaN
4                  10.6                        27         244.0
```

تابع describe: از این تابع برای مشاهده برخی اطلاعات آماری دیتای وارد شده استفاده می‌شود.

همان‌طور که در خروجی زیر دیده می‌شود اطلاعاتی مانند میانگین، انحراف معیار، مینیموم و ماکسیمم داده‌ها، چارک اول و دوم و سوم و... در خروجی این تابع ارائه می‌شود.(چند ستون اول خروجی آورده شده است)

	Unnamed: 0	MODELYEAR	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_CITY	\
count	1067.000000	1067.0	1040.000000	1033.000000	1067.000000	
mean	533.000000	2014.0	3.324038	5.797677	13.296532	
std	308.160672	0.0	1.411400	1.807262	4.101253	
min	0.000000	2014.0	1.000000	3.000000	4.600000	
25%	266.500000	2014.0	2.000000	4.000000	10.250000	
50%	533.000000	2014.0	3.300000	6.000000	12.600000	
75%	799.500000	2014.0	4.200000	8.000000	15.550000	
max	1066.000000	2014.0	8.400000	12.000000	30.200000	

تابع tail: این تابع ردیف‌های پایین یک DataFrame یا سری را برمی گرداند که n مقدار ورودی کاربر است. در این بخش ما برای tail ورودی تعیین نکردیم در نتیجه فقط ۵ تا از ردیف‌ها را بر می‌گرداند.(مقدار دیفالت آن ۵ ردیف است)

خروجی این بخش:

	Unnamed: 0	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	\
1062	1062	2014	VOLVO	XC60 AWD	SUV - SMALL	3.0	
1063	1063	2014	VOLVO	XC60 AWD	SUV - SMALL	3.2	
1064	1064	2014	VOLVO	XC70 AWD	SUV - SMALL	3.0	
1065	1065	2014	VOLVO	XC70 AWD	SUV - SMALL	3.2	
1066	1066	2014	VOLVO	XC90 AWD	SUV - STANDARD	3.2	

	CYLINDERS	TRANSMISSION	FUELTYPE	FUELCONSUMPTION_CITY	\
1062	6.0	AS6	X	13.4	
1063	6.0	AS6	X	13.2	
1064	6.0	AS6	X	13.4	
1065	6.0	AS6	X	12.9	
1066	6.0	AS6	X	14.9	

	FUELCONSUMPTION_HWY	FUELCONSUMPTION_COMB	FUELCONSUMPTION_COMB_MPG	\
1062	9.8	11.8	24	
1063	9.5	11.5	25	
1064	9.8	11.8	24	
1065	9.3	11.3	25	
1066	10.2	12.8	22	

بخش دوم:

با استفاده از تابع info کتابخانه pandas نوع هر کدام ستون‌های داده نشان داده شده است. بعضی ستون ها از نوع دسته‌ای و بعضی دیگر از نوع عددی هستند.

```
fuelData.info(verbose=True)
✓ 0.4s
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1067 entries, 0 to 1066
Data columns (total 14 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Unnamed: 0                1067 non-null   int64
 1   MODELYEAR                 1067 non-null   int64
 2   MAKE                      1067 non-null   object
 3   MODEL                     1058 non-null   object
 4   VEHICLECLASS              1067 non-null   object
 5   ENGINESIZE                1040 non-null   float64
 6   CYLINDERS                 1033 non-null   float64
 7   TRANSMISSION              1067 non-null   object
 8   FUELTYPE                  996 non-null    object
 9   FUELCONSUMPTION_CITY      1067 non-null   float64
 10  FUELCONSUMPTION_HWY       1067 non-null   float64
```

در قسمت بعد ستون دسته‌ای با نام FUELTYPE را که شامل مقادیر X ,E ,D, Z می باشد، به گونه ای تغییر دادیم که هر کدام از این مدل‌ها به یکی از اعداد بازه‌ی صفر تا سه نگاشت شوند.

```
fuelData['FUELTYPE'] = fuelData['FUELTYPE'].astype('category')
fuelData['FUELTYPE'] = fuelData['FUELTYPE'].cat.codes
fuelData.tail(n = 50)
```

خروجی این بخش به صورت زیر است. ۵۰ ردیف آخر چاپ شده است و همان‌طور که در ستون FUELTYPE مشاهده می‌کنید FUELTYPE به یکی از اعداد ۰و۱و۲و۳ نگاشت شده است. برای این کار از تابع astype برای تبدیل ستون به نوع دسته‌ای استفاده کرده‌ایم. سپس از cat.codes برای نگاشت FUELTYPE استفاده می‌نماییم. این تابع هر یک از نوع‌های FUELTYPE را به یک عدد نگاشت می‌کند(منفی یک برای داده‌ی Nan است) در صفحه‌ی بعد داده‌ی اصلی را مشاهده می‌نمایید. مثلا سوخت Z به عدد ۳ مپ شده است.

```
fuelData['FUELTYPE'] = fuelData['FUELTYPE'].astype('category')
fuelData['FUELTYPE'] = fuelData['FUELTYPE'].cat.codes
fuelData.tail(n = 50)
✓ 0.8s                                                                    Python
```

	Unnamed: 0	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION	FUELTYPE	F
1017	1017	2014	VOLKSWAGEN	BEETLE	COMPACT	2.0	4.0	M6	3	
1018	1018	2014	VOLKSWAGEN	BEETLE	COMPACT	2.5	5.0	A6	2	
1019	1019	2014	VOLKSWAGEN	BEETLE	COMPACT	2.5	5.0	M5	2	
1020	1020	2014	VOLKSWAGEN	BEETLE CONVERTIBLE	SUBCOMPACT	1.8	4.0	A6	-1	
1021	1021	2014	VOLKSWAGEN	BEETLE CONVERTIBLE	SUBCOMPACT	2.0	4.0	A6	3	
1022	1022	2014	VOLKSWAGEN	BEETLE CONVERTIBLE	SUBCOMPACT	2.0	4.0	M6	3	
1023	1023	2014	VOLKSWAGEN	BEETLE CONVERTIBLE	SUBCOMPACT	2.5	5.0	A6	2	
1024	1024	2014	VOLKSWAGEN	BEETLE TDI CLEAN DIESEL	COMPACT	2.0	4.0	A6	0	

	Unnamed: 0	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION	FUELTYPE	F
1017	1017	2014	VOLKSWAGEN	BEETLE	COMPACT	2.0	4.0	M6	Z	
1018	1018	2014	VOLKSWAGEN	BEETLE	COMPACT	2.5	5.0	A6	X	
1019	1019	2014	VOLKSWAGEN	BEETLE	COMPACT	2.5	5.0	M5	X	
1020	1020	2014	VOLKSWAGEN	BEETLE CONVERTIBLE	SUBCOMPACT	1.8	4.0	A6	NaN	
1021	1021	2014	VOLKSWAGEN	BEETLE CONVERTIBLE	SUBCOMPACT	2.0	4.0	A6	Z	
1022	1022	2014	VOLKSWAGEN	BEETLE CONVERTIBLE	SUBCOMPACT	2.0	4.0	M6	Z	
1023	1023	2014	VOLKSWAGEN	BEETLE CONVERTIBLE	SUBCOMPACT	2.5	5.0	A6	X	
1024	1024	2014	VOLKSWAGEN	BEETLE TDI CLEAN DIESEL	COMPACT	2.0	4.0	A6	D	
1025	1025	2014	VOLKSWAGEN	BEETLE TDI CLEAN DIESEL	COMPACT	2.0	4.0	M6	D	

بخش سوم:

ابتدا با استفاده از کد زیر برای هر ستون تعداد سطرهایی را که مقدار آن ستون برای آنها خالی است نشان می‌دهیم.

ابتدا با استفاده از len مقدار طول دیتافریم را بدست می‌آوریم و سپس تعداد دیتاهای پر را از آن کم می‌کنیم. تعداد دیتاهای خالی بدست می‌آید.

```
count_nan = len(fuelData) - fuelData.count()
print(count_nan)
```

✓ 0.4s

```
Unnamed: 0                  0
MODELYEAR                   0
MAKE                        0
MODEL                       9
VEHICLECLASS                0
ENGINESIZE                 27
CYLINDERS                  34
TRANSMISSION                0
FUELTYPE                   71
FUELCONSUMPTION_CITY        0
FUELCONSUMPTION_HWY         0
FUELCONSUMPTION_COMB        0
FUELCONSUMPTION_COMB_MPG    0
CO2EMISSIONS              103
```

سپس با استفاده از تکه کد زیر مقدار داده‌های عددی را با میانگینشان و داده‌های دسته‌ای را با مدشان پر می‌نماییم.

```
numeric_columns = fuelData.select_dtypes(include=['number']).columns.difference(['CO2EMISSIONS'])
fuelData[numeric_columns] = fuelData[numeric_columns].fillna(fuelData.mean())

Categorical_columns = fuelData.select_dtypes(exclude=['number']).columns
fuelData[Categorical_columns] = fuelData[Categorical_columns].transform(lambda a: a.fillna(a.mode()[0]))
```

در خط اول داده‌های عددی (به جز ستون هدف) را جدا کرده، سپس مقدار میانگین را در مقادیر Nan این ستون‌ها قرار می‌دهیم.

در خط بعد داده‌های دسته‌ای را جدا می‌کنیم و مقدار مد را در مقادیر Nan آن جایگذاری می‌کنیم. چون بعضی از داده‌ها چند مد داشتند، در داخل پرانتز تاکید کردیم که اولین مد را برای هر کدام بریزد.

سپس دوباره تعداد خانه‌های Nan را حساب می‌کنیم. همان‌طور که می‌بینید جز ستون هدف بقیه‌ی خانه‌ها پر شده‌اند.

```
Unnamed: 0                    0
MODELYEAR                     0
MAKE                          0
MODEL                         0
VEHICLECLASS                  0
ENGINESIZE                    0
CYLINDERS                     0
TRANSMISSION                  0
FUELTYPE                      0
FUELCONSUMPTION_CITY          0
FUELCONSUMPTION_HWY           0
FUELCONSUMPTION_COMB          0
FUELCONSUMPTION_COMB_MPG      0
CO2EMISSIONS                103
```

مزایای این کار:

این روش زمانی مناسب است که حجم داده ها کوچک باشد.

معایب این کار:

این روش واریانس مجموعه داده را تغییر می دهد و در نتیجه ما برآورد کمتری نسبت به واریانس دیتای واقعی داریم.

در مقایسه با سایر روش‌ها ضعیف عمل می کند.

یکی دیگر از معایب احتمالی استفاده از میانگین برای مقادیر از دست رفته این است که دلیل از دست رفتن مقادیر در مرحله اول می تواند به خود مقادیر گم شده بستگی داشته باشد. مثلا اگر در سطح شهر بخواهیم یک آزمایش در مورد سلامتی افراد بگیریم افرادی که از سلامتی کمتری برخوردارند به علت عدم علاقه به ابراز آن در آزمایش شرکت نمی‌کنند و اگر ما میانگین را در مقادیر از دست رفته جایگذاری بکنیم پیش‌بینی مناسبی انجام نداده‌ایم.

در آخر این بخش نیز سطرهایی که مقدار ستون هدف آنها NaN است را از دیتافریم اصلی جدا کرده و در دیتافریم جدیدی ذخیره می‌کنیم.

با استفاده از تکه کد زیر:

```
NanNumber = fuelData['CO2EMISSIONS'].isnull()
NotNanNumber = ~fuelData['CO2EMISSIONS'].isnull()
fuelDataNan = fuelData[NanNumber]
fuelDataNotNan = fuelData[NotNanNumber]
fuelDataNan
fuelDataNotNan
```

خروجی تکه کد بالا دو دیتافریم است که اندازه یکی ۱۴*۱۰۳ است و اندازه دیگری ۱۴*۹۶۴ است.

آن دیتافریمی که ۱۰۳ ردیف دارد همان است که دارای داده‌های Nan می‌باشد و داده‌ی تست ما می‌باشد. (شکل پایین)

	Unnamed: 0	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION	FUELTYPE
3	3	2014	ACURA	MDX 4WD	SUV - SMALL	3.5	6.000000	AS6	4
20	20	2014	AUDI	A4 QUATTRO	COMPACT	2.0	4.000000	AS8	4
30	30	2014	AUDI	A8	MID-SIZE	3.0	6.000000	AS8	0
42	42	2014	AUDI	Q7	SUV - STANDARD	3.0	5.797677	AS8	4
43	43	2014	AUDI	Q7 TDI CLEAN DIESEL	SUV - STANDARD	3.0	6.000000	AS8	1
...	...	...	...	...	...	...	...	...	...
1022	1022	2014	VOLKSWAGEN	BEETLE CONVERTIBLE	SUBCOMPACT	2.0	4.000000	M6	4
1027	1027	2014	VOLKSWAGEN	CC	COMPACT	2.0	4.000000	M6	4
1051	1051	2014	VOLKSWAGEN	TIGUAN	SUV - SMALL	2.0	4.000000	A6	4
1052	1052	2014	VOLKSWAGEN	TIGUAN	SUV - SMALL	2.0	4.000000	M6	4
1053	1053	2014	VOLKSWAGEN	TIGUAN 4MOTION	SUV - SMALL	2.0	4.000000	A6	4

103 rows × 14 columns

دیتافریم زیر نیز داده‌ی ترین ما می‌باشد.

	Unnamed: 0	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION	FUELTYPE	FUELCONS
0	0	2014	ACURA	ILX	COMPACT	2.0	4.0	AS5	4	
1	1	2014	ACURA	ILX	COMPACT	2.4	4.0	M6	4	
2	2	2014	ACURA	ILX HYBRID	COMPACT	1.5	4.0	AV7	4	
4	4	2014	ACURA	RDX AWD	SUV - SMALL	3.5	6.0	AS6	4	
5	5	2014	ACURA	RLX	MID-SIZE	3.5	6.0	AS6	4	
...	...	...	...	...	...	...	...	...	...	
1062	1062	2014	VOLVO	XC60 AWD	SUV - SMALL	3.0	6.0	AS6	3	
1063	1063	2014	VOLVO	XC60 AWD	SUV - SMALL	3.2	6.0	AS6	3	
1064	1064	2014	VOLVO	XC70 AWD	SUV - SMALL	3.0	6.0	AS6	3	
1065	1065	2014	VOLVO	XC70 AWD	SUV - SMALL	3.2	6.0	AS6	3	
				XC90	SUV -					

بخش چهار:

با استفاده از تکه کد زیر میانگین‌های مورد نظر این بخش محاسبه شد.

```
goal_columns2 = fuelData['CO2EMISSIONS'] < 240
fuelData2 = fuelData.groupby(goal_columns2)
meanFuel2 = fuelData2['FUELCONSUMPTION_CITY'].mean()
print(meanFuel2[1])

goal_columns3 = fuelData['CO2EMISSIONS'] > 300
fuelData3 = fuelData.groupby(goal_columns3)
meanFuel3 = fuelData3['FUELCONSUMPTION_CITY'].mean()
print(meanFuel3[1])
```

✓ 0.7s

```
10.037819025522042
18.663255813953487
```

همان‌طور که مشاهده می‌شود مقدار این میانگین برای قسمت اول که کربن‌دی‌اکسید خروجیشان کمتر از ۲۴۰ بود ۱۰.۰۳ شد و برای قسمت بعد که کربن‌دی‌اکسید خروجیشان بیشتر از ۳۰۰ بود ۱۸.۶۶ می‌باشد.

بخش پنج:

در این قسمت با روش حلقه میانگین را حساب می‌کنیم. خروجی مانند بخش قبل است.

```
count1 = 0
sum1 = 0
for i in range(len(fuelData)):
    if fuelData['CO2EMISSIONS'][i] < 240:
        sum1 += fuelData['FUELCONSUMPTION_CITY'][i]
        count1 += 1
print(sum1 / count1)

count2 = 0
sum2 = 0
for i in range(len(fuelData)):
    if fuelData['CO2EMISSIONS'][i] > 300:
        sum2 += fuelData['FUELCONSUMPTION_CITY'][i]
        count2 += 1

print(sum2 / count2)
```

✓ 0.6s

```
10.037819025522042
18.663255813953487
```

حال به مقایسه‌ی زمان‌های این دو راه حل می‌پردازیم.

برای محاسبه زمان صرف‌شده برای این دو از کتابخانه time کمک گرفته شده است به طوری که اول هر عملیات زمان شروع و پایان عملیات‌ها زمان پایان ثبت شده است و در نهایت از یکدیگر مقدارشان را کم می‌کنیم.

```
print(endTime1 - startTime1)
print(endTime2 - startTime2)
```

✓ 0.5s

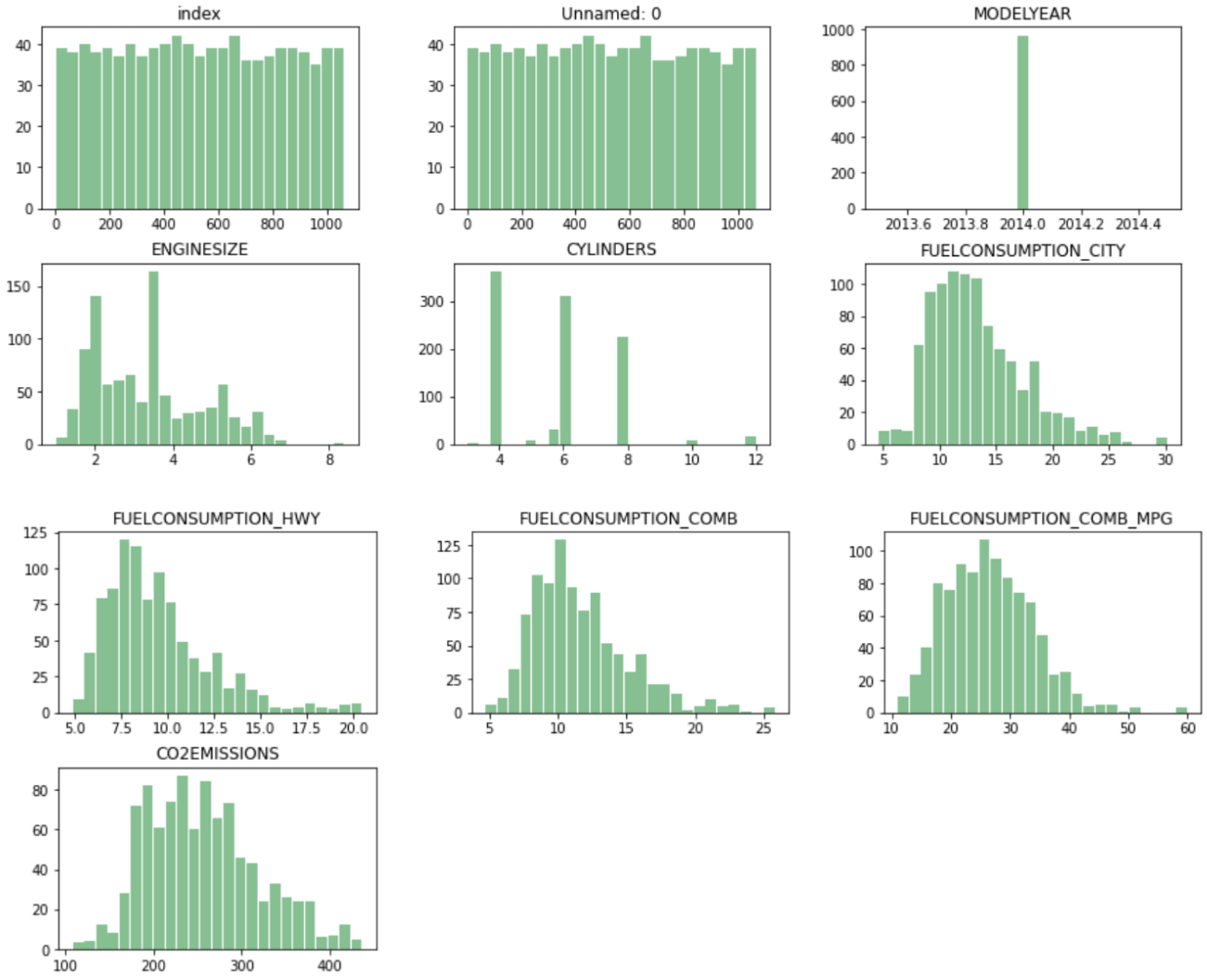
```
0.012742757797241211
0.03741002082824707
```

همان‌طور که مشاهده می‌شود زمان عملیات با حلقه چند برابر عملیات با vectorization است.

بخش شش:

```
fuelDataNotNan.hist(bins=25, grid=False, figsize=(15,12), color='#86bf92', zorder=2, rwidth=0.9)
```

نمودار هیستوگرام هر ستون از داده را مشاهده می‌نمایید. دو نمودار اول مربوط به ایندکس‌ها می‌باشند.



بخش هفت:

نتیجه نرمال‌سازی داده‌ها:

```
newNumericFuelData = (fuelDataNotNan2 - fuelDataNotNan2.mean()) / fuelDataNotNan2.std()
newNumericFuelData
```

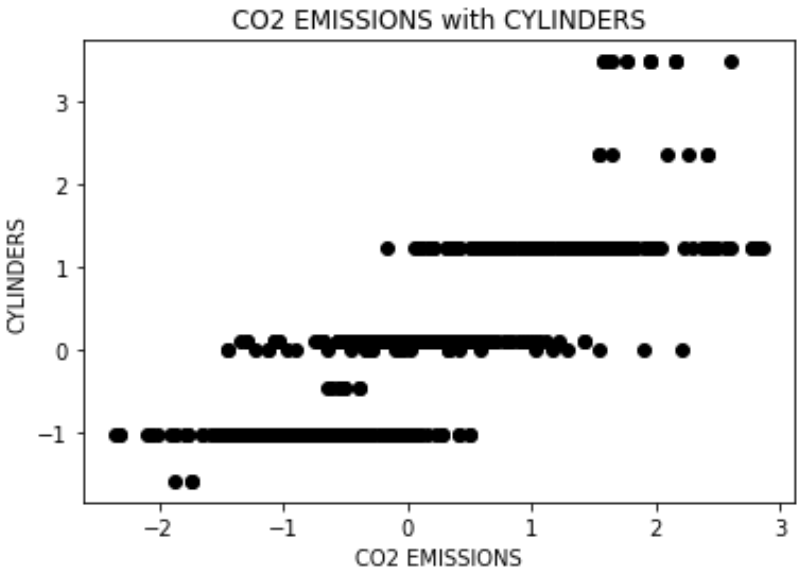
✓ 0.7s Python

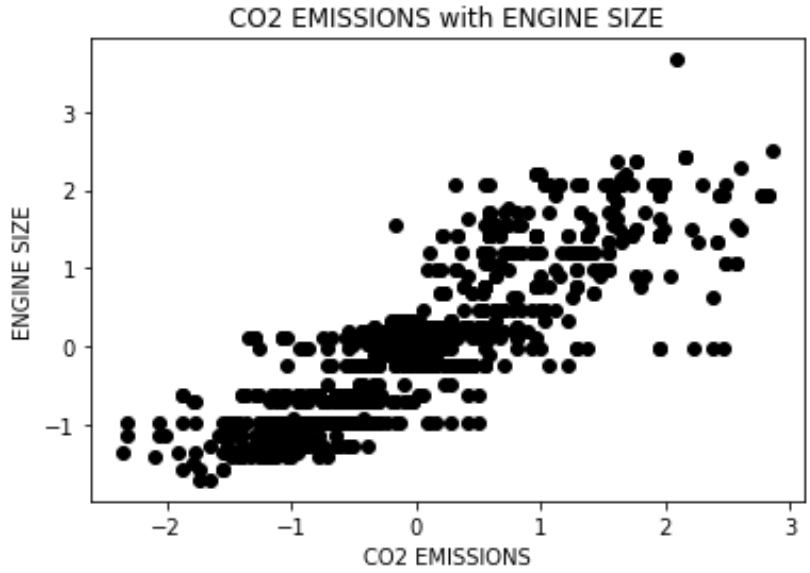
	CO2EMISSIONS	CYLINDERS	ENGINESIZE	FUELCONSUMPTION_CITY	FUELCONSUMPTION_COMB	FUELCONSUMPTION_COMB_MPG
0	-0.960111	-1.024099	-0.973622	-0.833605	-0.888216	0.88628
1	-0.564949	-1.024099	-0.683744	-0.519842	-0.576247	0.35288
2	-1.908498	-1.024099	-1.335970	-1.774893	-1.625596	2.88655
3	-0.201401	0.100355	0.113422	-0.302621	-0.292640	0.08618
4	-0.422691	0.100355	0.113422	-0.350893	-0.462804	0.21953
...	...	...	...	...	...	...
959	0.225373	0.100355	-0.248926	0.011141	0.047690	-0.31387
960	0.114728	0.100355	-0.103987	-0.037130	-0.037393	-0.18052
961	0.225373	0.100355	-0.248926	0.011141	0.047690	-0.31387
962	0.051502	0.100355	-0.103987	-0.109537	-0.094114	-0.18052
963	0.588921	0.100355	-0.103987	0.373175	0.331297	-0.58057

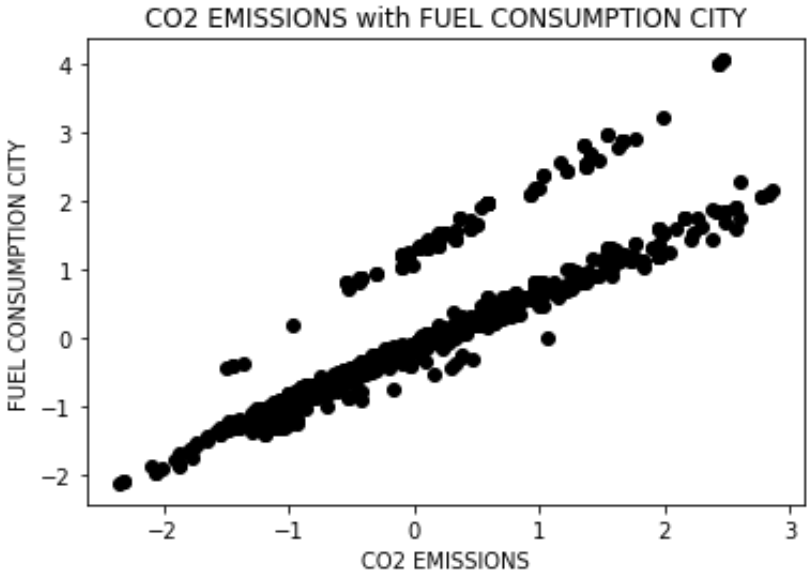
964 rows × 10 columns

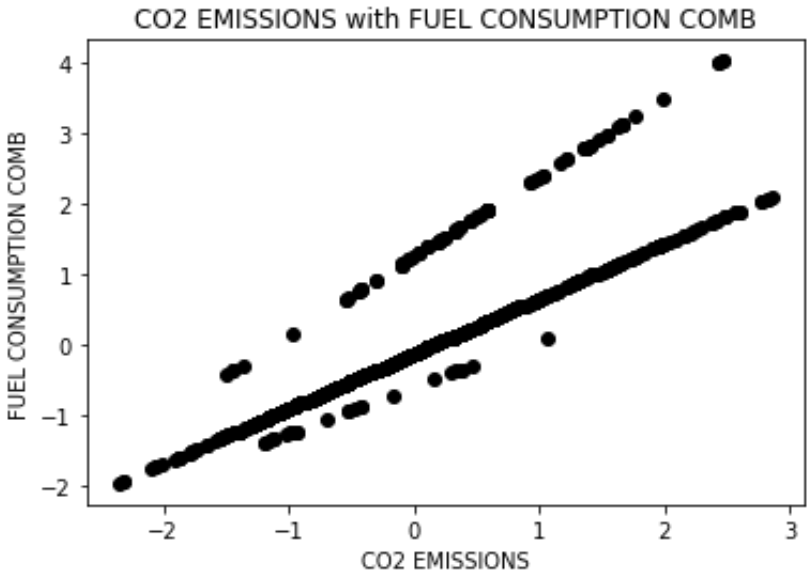
بخش هشت:

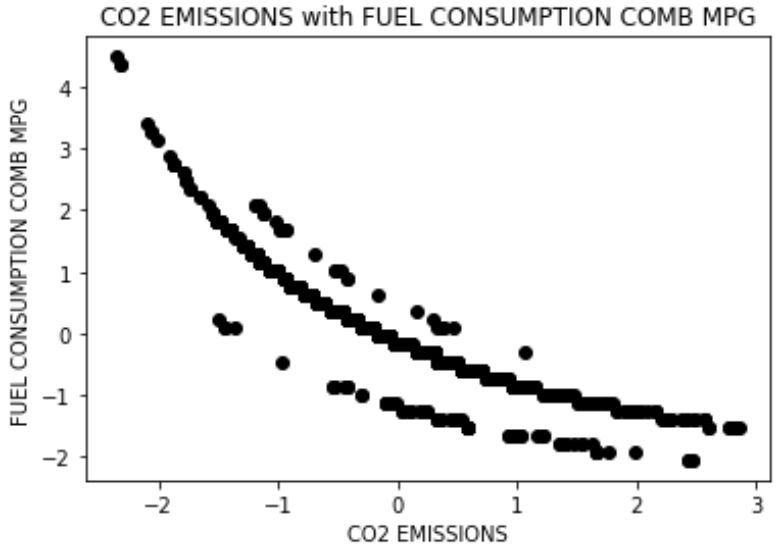
الف) با استفاده از تابع matplot.pyplot.scatter نمودارها را رسم می‌نماییم.

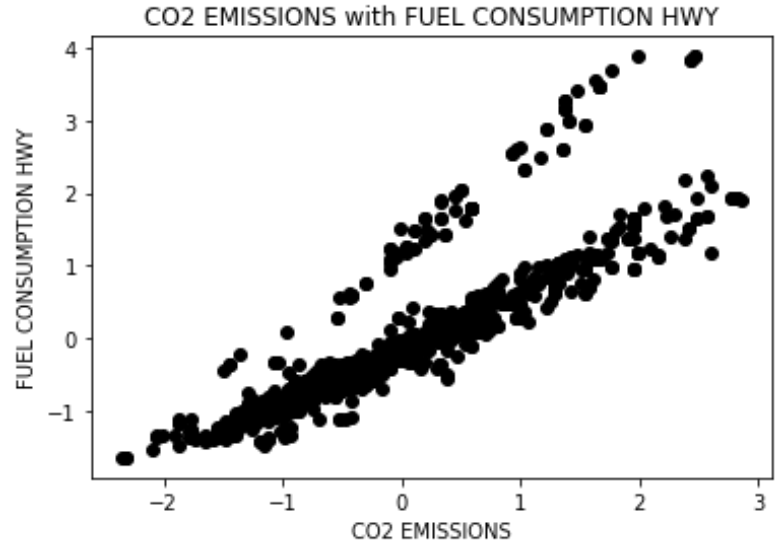


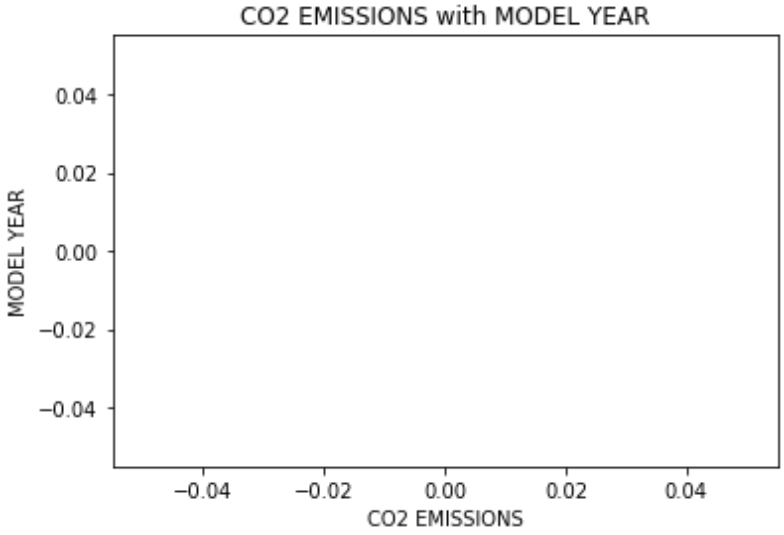

CO2 EMISSIONS with ENGINE SIZE
ENGINE SIZE
CO2 EMISSIONS
3
2
1
0
−1
−2
−1
0
1
2
3


CO2 EMISSIONS with FUEL CONSUMPTION CITY
FUEL CONSUMPTION CITY
CO2 EMISSIONS
4
3
2
1
0
−1
−2
−2
−1
0
1
2
3


CO2 EMISSIONS with FUEL CONSUMPTION COMB
FUEL CONSUMPTION COMB
CO2 EMISSIONS
4
3
2
1
0
−1
−2
−2
−1
0
1
2
3







از آن جایی که از داده نرمالایز شده برای کشیدن پلات‌ها استفاده کرده‌ایم و مقادیر نرمالایز MODELYEAR ، NAN شده‌اند(به علت صفر بودن انحراف معیار) در نتیجه پلات آن نیز خالی است.

ب) ویژگی FUELCONSUMPTION_COMB بیشترین همبستگی را دارد زیرا می‌بینیم نسبت به بقیه‌ی ویژگی‌ها نسبت خطی‌اش واضح‌تر است. به صورت منطقی هم که فکر بکنیم میزان مصرف کربن دی‌اکسید با نسبتی که سوخت مصرف می‌شود رابطه دارد و این میزان ارتباط در جمع سوخت شهر و اتوبان واضح‌تر است.

بخش نه: داده‌ی جدید توسط خط زیر جدا شد.

```
fuelDataNew = fuelDataNotNan2[['CO2EMISSIONS', 'FUELCONSUMPTION_COMB']]
```

بخش ده: تابع تخمین‌گر توسط تکه کد زیر نوشته شد.

```
A = np.arange(1*2.00).reshape(2, 1)
X = np.arange(len(fuelDataNew)*2.00).reshape(len(fuelDataNew), 2)
X[:, 0] =  1
X[:, 1] = fuelDataNew['FUELCONSUMPTION_COMB']
Y = fuelDataNew['CO2EMISSIONS']
X_pinv = np.linalg.pinv(X)
A = np.dot(X_pinv,Y)
A
```

✓ 0.2s

```
array([1.99493200e-17, 8.88687103e-01])
```

ابتدا آرایه ایکس که یک آرایه‌ی طول دیتا ضربدر دو است ساخته می‌شود که ستون اول آن را عدد یک تشکیل می‌دهد و ستون دوم آن اطلاعات FUELCONSUMPTION_COMB است. معکوس این ماتریس در ایگرگ که همان اطلاعات CO2EMMISION است، ضرب می‌شود و حاصل این ضرب یک ماتریس دو در یک است که ضرایب معادله‌ی تخمین‌گر ما می‌باشد. عضو اول آرایه عرض از مبدا و عضو دوم آن شیب می‌باشد.

حال با توجه به این مقادیر MSE را محاسبه می‌نماییم.

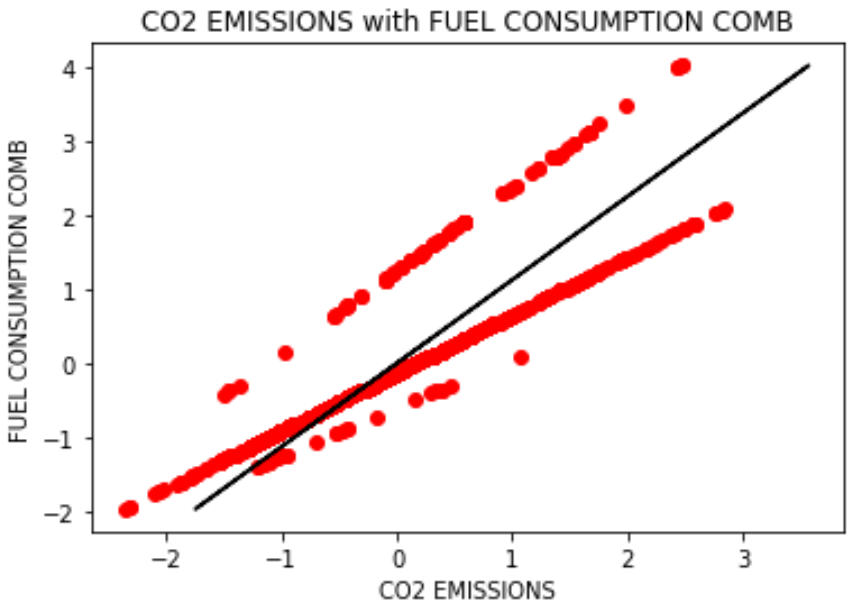
```
H = A[1]*fuelDataNew['FUELCONSUMPTION_COMB'] + A[0]
MSE = sum((fuelDataNew['CO2EMISSIONS'] - H)**2)/len(fuelDataNew)
print(MSE)
```

✓ 0.3s

```
0.21001714701294727
```

مشاهده می‌شود که مقدار خطا از ۰.۵ کمتر است پس داده‌ای که انتخاب کردیم مناسب می‌باشد.

بخش یازده:



پلات قرمز رنگ نشان‌دهنده رابطه بین CO2EMMISION و FUELCONSUMPTION_COMB است و خط سیاه، رابطه بین حدسی که ما زدیم و FUELCONSUMPTION_COMB است.

از آن جایی که ویژگی مورد نظر با CO_2 همبستگی دارد پس تقریبی که زدیم هم تقریبا با واقعیت همخوانی دارد و به همین علت است که این دو نمودار تقریبا روی هم می‌افتند.

بخش دوازده:

نتیجه‌های بدست آمده از طریق تابع تخمین‌گر را در عکس پایین مشاهده می‌کنید.مقادیر همان‌طور که حدس زده می‌شد بین ۲۰۰ تا ۳۰۰ هستند.

```
fuelDataNan['CO2EMISSIONS'] = (A2[1]*fuelDataNan['FUELCONSUMPTION_COMB'] + A2[0])
```

	CO2EMISSIONS
3	248.261254
20	230.721434
30	249.855783
42	281.746366
43	245.072196
...	...
1022	221.154259
1027	227.532375
1051	241.883138
1052	256.233900
1053	241.883138

نتیجه‌گیری کلی: با توجه به این پروژه دانستیم که می‌توان با انتخاب یک متغیر همبسته مقدار خانه‌های گمشده یک دیتا را با تقریب خوبی پر کرد.

ارائه راهکارهایی برای توسعه و بهبود پروژه:

اول خسته نباشید عرض می‌کنم بابت این پروژه مفید. پروژه در ابتدا بسیار سخت بنظر می‌آمد ولی کم کم دستمان در کد زدن تندتر شد و با قواعد آشناتر شدیم که بنظرم ویژگی خوبی بود. ولی در کل بنظرم برای پروژه صفر حجم زیادی داشت :D

منابع استفاده شده:

https://stackoverflow.com

https://pandas.pydata.org

https://www.geeksforgeeks.org

https://analyticsindiamag.com/5-ways-handle-missing-values-machine-learning-datasets/

https://stackoverflow.com/questions/35077507/how-to-right-align-and-justify-align-in-markdown

https://www.pythontutorial.net/python-basics/python-write-csv-file/

https://stackoverflow.com/questions/1557571/how-do-i-get-time-of-a-python-programs-execution

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.std.html

https://stackoverflow.com/questions/46377331/solving-ax-b-for-a-non-square-matrix-a-using-python

https://www.geeksforgeeks.org/pandas-dataframe-hist-function-in-python/

https://towardsdatascience.com/an-easy-way-to-divide-your-dataset-based-on-data-types-with-pandas-4625411a57b

https://appdividend.com/2020/05/26/pandas-dataframe-head-method-in-python/#:~:text=Pandas%20DataFrame%20head()%20method%20returns%20top%20n%20rows%20of,type%20of%20data%20in%20it.

https://www.geeksforgeeks.org/python-pandas-dataframe-describe-method/#:~:text=Pandas%20describe()%20is%20used,shown%20in%20the%20examples%20below.&text=Return%20type%3A%20Statistical%20summary%20of%20data%20frame.

https://www.w3resource.com/pandas/dataframe/dataframe-tail.php#:~:text=The%20tail()%20function%20is,after%20sorting%20or%20appending%20rows.&text=Number%20of%20rows%20to%20select.

and …