HW2

March 4, 2020

1 HW2

Using the gpa data from the Data folder on GitHub (gpa.csv), build a predictive linear regression model using the sklearn package.

[1]:
```
import warnings
warnings.filterwarnings('ignore')



import pandas as pd
import numpy as np
from plotnine import *
import statsmodels.api as sm

from sklearn.linear_model import LinearRegression # Linear Regression Model
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import mean_squared_error, r2_score #model evaluation

from sklearn.model_selection import train_test_split # simple TT split cv
from sklearn.model_selection import KFold # k-fold cv
from sklearn.model_selection import LeaveOneOut #LOO cv
from sklearn.model_selection import cross_val_score # cross validation metrics
from sklearn.model_selection import cross_val_predict # cross validation metrics

%matplotlib inline
```

1.1 1

Use plotnine to explore the data (what patterns do you see in the data? Are any of them surprising?)

- GPA is the grade point average of a student
- ParentsIncome is the income of the student's family
- SAT.Math, SAT.Reading, and SAT.Writing are the student's SAT scores
- PeanutAllergy is a binary variable indicating whether the student has (1) or does not have (0) a peanut allergy.

[2]:
```
# Explore
#---YOUR CODE HERE-----------------------
```

```
gpa = pd.read_csv('data/gpa.csv')

gpa.info()
gpa.isnull().sum()


#---/YOUR CODE HERE----------------------
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
Unnamed: 0       1000 non-null int64
GPA              1000 non-null float64
ParentsIncome    1000 non-null float64
SAT.Math         1000 non-null int64
SAT.Reading      1000 non-null int64
SAT.Writing      1000 non-null int64
PeanutAllergy    1000 non-null int64
dtypes: float64(2), int64(5)
memory usage: 54.8 KB
```

[2]:
```
Unnamed: 0       0
GPA              0
ParentsIncome    0
SAT.Math         0
SAT.Reading      0
SAT.Writing      0
PeanutAllergy    0
dtype: int64
```

[3]:
```
gpa['SAT.Score'] = gpa['SAT.Math'] + gpa['SAT.Reading'] + gpa['SAT.Writing']
gpa.head()
```

[3]:

	Unnamed: 0	GPA	ParentsIncome	SAT.Math	SAT.Reading	SAT.Writing	PeanutAllergy	SAT.Score
0	1	3.03	48555.69	510	527	571	0	1608
1	2	3.55	48779.43	623	593	639	0	1855
2	3	3.83	49708.23	485	592	623	0	1700
3	4	2.63	52874.02	648	689	738	0	2075
4	5	3.60	51052.65	653	476	720	0	1849

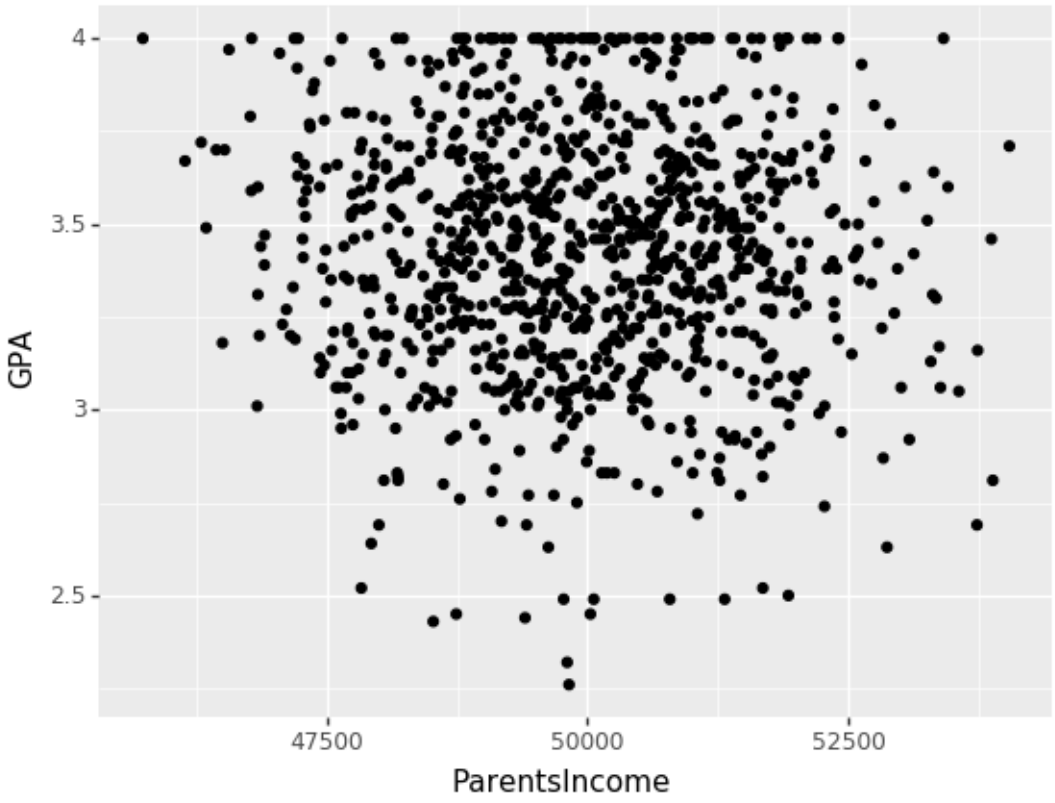
[4]:
```
gpa.describe()
```

[4]:

	Unnamed: 0	GPA	ParentsIncome	SAT.Math	SAT.Reading	\
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	
mean	500.500000	3.431990	49941.462410	601.034000	602.108000	
std	288.819436	0.331665	1457.996468	73.415943	75.730836	
min	1.000000	2.260000	45732.190000	366.000000	368.000000	
25%	250.750000	3.210000	48924.377500	554.000000	548.000000	
50%	500.500000	3.440000	49911.570000	601.000000	604.000000	
75%	750.250000	3.670000	50985.130000	650.250000	654.000000	
max	1000.000000	4.000000	54047.960000	800.000000	800.000000	

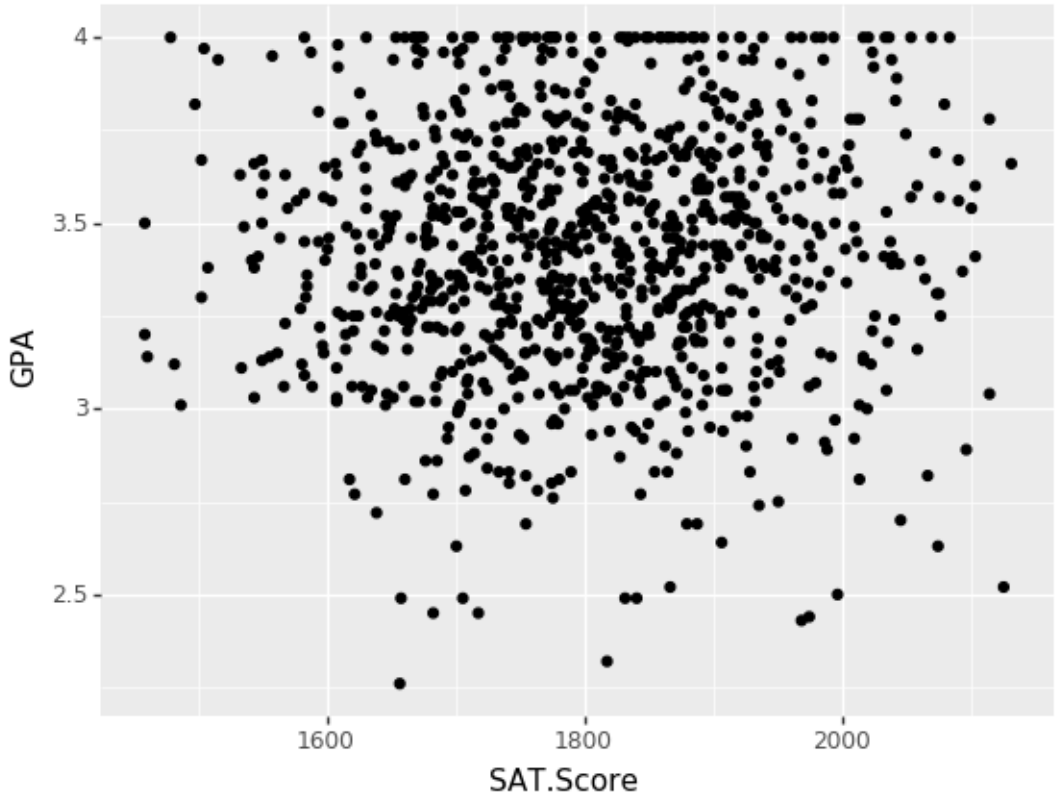
	SAT.Writing	PeanutAllergy	SAT.Score
count	1000.00000	1000.000000	1000.000000
mean	599.21100	0.024000	1802.353000
std	74.37859	0.153126	127.134415
min	372.00000	0.000000	1459.000000
25%	548.00000	0.000000	1709.750000
50%	600.00000	0.000000	1800.000000
75%	650.00000	0.000000	1889.250000
max	800.00000	1.000000	2132.000000

[5]:
```
(ggplot(gpa, aes(x = "ParentsIncome", y = "GPA"))
       +geom_point())
```



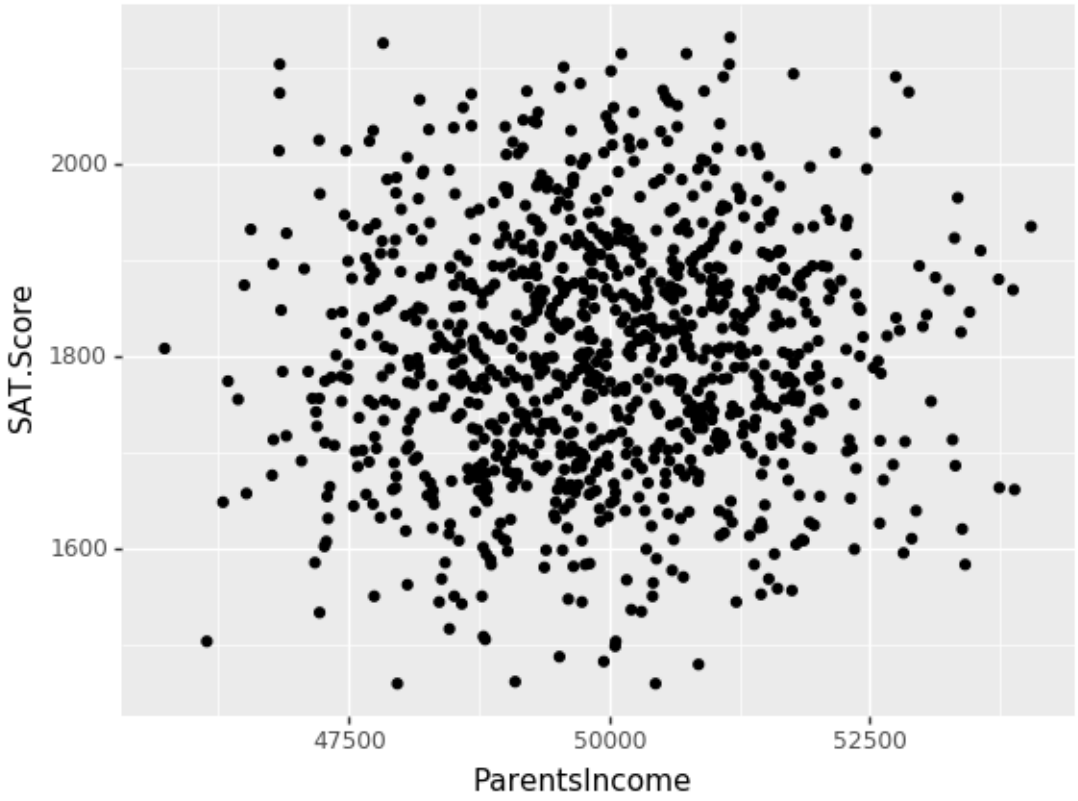
[5]: <ggplot: (-9223371934995276276)>

[6]:
```
(ggplot(gpa, aes(x = "SAT.Score", y = "GPA"))
        +geom_point())
```



[6]: <ggplot: (-9223371934994956700)>

[7]:
```
(ggplot(gpa, aes(x = "ParentsIncome", y = "SAT.Score"))
        +geom_point())
```



[7]: <ggplot: (-9223371934994953772)>

Describe patterns here: Data seems to be very scattered (heteroskedastic). I initially thought that parents income would have a larger impact on GPA and SAT scores, and I think it does so slightly around the $48.000 mark, however thought it would be much more drastic. Same goes for SAT Score which seems to be even more scattered.

1.2 2

Build a predictive linear regression model (using sklearn) that predicts GPA based on other variables. Why did you choose the predictor variables you did? Justify your answer. Make sure to standardize continuous variables.

[8]:
```
# Model

#---YOUR CODE HERE-------------------------
predictors = ["ParentsIncome", "PeanutAllergy", "SAT.Math", "SAT.Reading", "SAT.
 ↪Writing", "SAT.Score"]

X_train, X_test, y_train, y_test = train_test_split(gpa[predictors],␣
 ↪gpa["GPA"], test_size=0.2)
```

```
print('X_train is:', X_train.shape)
print('X_test is:', X_test.shape)
print('y_train is:', y_train.shape)
print('y_test is:', y_test.shape)

#---/YOUR CODE HERE----------------------
```

```
X_train is: (800, 6)
X_test is: (200, 6)
y_train is: (800,)
y_test is: (200,)
```

[9]:
```
zscore = StandardScaler()
zscore.fit(X_train)
Xz_train = zscore.transform(X_train)
Xz_test = zscore.transform(X_test)
```

Justify your predictor variable selection here: I chose SAT scores and its subject fields because scores are generally used to guage a students aptitude and capability, and I used parents income because it can represent a students economic backgroud, and I chose peanut allergy because it may also be representative of the type of environment and school system the student is in.

1.3 3

Check how your model did using the r^2 score and the mean squared error. How do you think your model did? Why do you think that?

[10]:
```
# Model Performance
#---YOUR CODE HERE-----------------------
model = LinearRegression()
model.fit(X_train, y_train)

train_pred = model.predict(X_train)
test_pred = model.predict(X_test)


print('training r2 is:', model.score(X_train, y_train)) #training R2
print('testing r2 is:', model.score(X_test, y_test)) #testing R2

print('\ntrain mse is: ', mean_squared_error(y_train,train_pred))
print('test mse is: ', mean_squared_error(y_test,test_pred))
#---/YOUR CODE HERE----------------------
```

```
training r2 is: 0.0028625259157192273
testing r2 is: 0.0011906966485256687

train mse is:  0.11084671859757499
```

```
test mse is:  0.10414588601283035
```

Describe your model performance and interpret it: Our OLS regression model acounts for approximately .4% of variance within our training dataset and .1% in our testing set, meaning that are model does a very poor job in predicting GPA. MSE values are fairly close with testing mse being slightly higher, potentially signalling slight overfitting, but not by much.

1.4 4

Interpret each coefficient from the model. What does each one mean in the context of this problem?

[11]:
```
# Coefficients
#---YOUR CODE HERE-----------------------
coefficients = pd.DataFrame({"Coef":model.coef_,
              "Name": predictors})

coefficients = coefficients.append({"Coef": model.intercept_,
               "Name": "intercept"}, ignore_index = True)

coefficients

#---/YOUR CODE HERE-----------------------
```

[11]:

	Coef	Name
0	-0.000010	ParentsIncome
1	0.003403	PeanutAllergy
2	0.000083	SAT.Math
3	-0.000015	SAT.Reading
4	-0.000021	SAT.Writing
5	0.000047	SAT.Score
6	3.848947	intercept

Interpret your coefficients here: With an increase of one standard deviation in each corresponding coefficient, GPA shifts in the unit of standard deviations according to the coefficient value — assuming other variables are held constant.

ParentsIncome - 1 SD change causes GPA to go down by -.00001: money makes you complacent and do less coursework? PeanutAllergy - 1 SD change causes GPA to go up by .003: Kids with allergies are going to better private schools? SAT.Math - 1 SD change causes GPA to go up by .00008: Math helps you in different subjects? SAT.Reading - 1 SD change causes GPA to go down by -.00002: Kids underestimate the math section? SAT.Writing - 1 SD change causes GPA to go down by -.00002: Kids who focus more on this subject neglect other areas? SAT.Score - 1 SD change causes GPA to go up by .00005: kids who are more well rounded have better a gpa? intercept - If all variables were 0, a students GPA would be predicted to be 3.8

[ ]: