# Import

```
#project by David Downing and Thoa Nguyen
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import plotly.express as px
import plotly as pt
import xgboost as xgb
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import mean_squared_error
from statistics import mean
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from numpy import absolute
```

```
df = pd.read_csv("https://drive.google.com/uc?export=download&id=1mZ0mVdp5KGXcoJExtsAaaQDPaOm
```

```
df.head()
```

| | Gender | Teacher | Period | Student ID | Gender.1 | Ethnicity | Economic | LEP | SCICOUR | CTE | . |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | M | Lewis | 4.0 | 43954 | 1 | 0 | 0.0 | 0.0 | 1.5 | 0.0 | |
| 1 | F | Howell | 6.0 | 47436 | 0 | 0 | 0.0 | 0.0 | 3.5 | 0.0 | |
| 2 | m | Lewis | 4.0 | 59755 | 0 | 0 | 0.0 | 0.0 | 1.5 | 1.5 | |
| 3 | M | Marshall | 5.0 | 35449 | 1 | 1 | 0.0 | 0.0 | 3.5 | 1.5 | |
| 4 | M | Lewis | 3.0 | 43956 | 1 | 1 | 0.0 | 0.0 | 1.5 | 0.5 | |

5 rows × 32 columns

# Cleaning

```
df.describe()
```

| | Period | Economic | LEP | CTE | BIO | ELA | Alg |
|---|---|---|---|---|---|---|---|
| **count** | 1279.000000 | 440.000000 | 440.000000 | 438.000000 | 425.000000 | 423.000000 | 418.000000 |
| **mean** | 4.517592 | 0.411364 | 0.115909 | 0.760868 | 81.402353 | 80.529551 | 79.966507 |
| **std** | 2.063422 | 4.252440 | 1.166238 | 1.042858 | 13.445400 | 10.726388 | 15.162709 |
| **min** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 38.000000 | 25.000000 | 9.000000 |
| **25%** | 3.000000 | 0.000000 | 0.000000 | 0.000000 | 74.000000 | 74.000000 | 70.250000 |
| **50%** | 4.000000 | 0.000000 | 0.000000 | 0.500000 | 84.000000 | 82.000000 | 83.000000 |
| **75%** | 6.000000 | 0.000000 | 0.000000 | 1.000000 | 92.000000 | 88.000000 | 93.000000 |
| **max** | 8.000000 | 89.000000 | 24.000000 | 5.000000 | 100.000000 | 100.000000 | 100.000000 |

8 rows × 25 columns

```
df.info() #got some categoricals and some continuous
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1295 entries, 0 to 1294
Data columns (total 32 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Gender      1260 non-null   object
 1   Teacher     1290 non-null   object
 2   Period      1279 non-null   float64
 3   Student ID  443 non-null    object
 4   Gender.1    441 non-null    object
 5   Ethnicity   441 non-null    object
 6   Economic    440 non-null    float64
 7   LEP         440 non-null    float64
 8   SCICOUR     439 non-null    object
 9   CTE         438 non-null    float64
 10  TOT         438 non-null    object
 11  BIO         425 non-null    float64
 12  ELA         423 non-null    float64
 13  Alg         418 non-null    float64
 14  GPA         440 non-null    float64
 15  TOSLS1      418 non-null    float64
 16  TOSLS2      418 non-null    float64
 17  TOSLS3      418 non-null    float64
 18  TOSLS4      418 non-null    float64
 19  TOSLS5      418 non-null    float64
 20  TOSLS6      418 non-null    float64
 21  TOSLS7      418 non-null    float64
 22  TOSLS8      418 non-null    float64
 23  TOSLS9      418 non-null    float64
 24  TOSLSTOT    418 non-null    float64
 25  BRAINS1     450 non-null    float64
```

```
 26  BRAINS2     450 non-null    float64
 27  BRAINS3     450 non-null    float64
 28  BRAINS4     450 non-null    float64
 29  BRAINS5     450 non-null    float64
 30  BRAINSTOT   450 non-null    float64
 31  SORT        1280 non-null   float64
dtypes: float64(25), object(7)
memory usage: 323.9+ KB
```

```
r,c = df.shape
r,c
```

```
(1295, 32)
```

```
df.isnull().sum() #some nulls but not so many we can't drop them and have plenty of observati
```

```
Gender         35
Teacher         5
Period         16
Student ID    852
Gender.1      854
Ethnicity     854
Economic      855
LEP           855
SCICOUR       856
CTE           857
TOT           857
BIO           870
ELA           872
Alg           877
GPA           855
TOSLS1        877
TOSLS2        877
TOSLS3        877
TOSLS4        877
TOSLS5        877
TOSLS6        877
TOSLS7        877
TOSLS8        877
TOSLS9        877
TOSLSTOT      877
BRAINS1       845
BRAINS2       845
BRAINS3       845
BRAINS4       845
BRAINS5       845
BRAINSTOT     845
SORT           15
dtype: int64
```

```
df2=df #creating a second cleaning that is slightly different
df2=df2.iloc[:438,:] #there is junk at the bottom of the csv, this is capturing the main colu
df2 = df2.dropna(subset=['TOSLSTOT']) # drop null values only for our target variable
```

```
df2=df2.reset_index()
df2=df2.drop([373])
df2['BIO'] = df2['BIO'].fillna((df2['BIO'].mean()))
df2['Alg'] = df2['Alg'].fillna((df2['Alg'].mean()))
df2['ELA'] = df2['ELA'].fillna((df2['ELA'].mean()))
df2['BRAINSTOT'] = df2['BRAINSTOT'].fillna((df2['BRAINSTOT'].mean()))
df2['Gender.1']=df2['Gender.1'].astype('float64') #these were type object for some reason
df2['Ethnicity']=df2['Ethnicity'].astype('float64')
df2['SCICOUR']=df2['SCICOUR'].astype('float64')
```

```
df2
```

| | index | Gender | Teacher | Period | Student ID | Gender.1 | Ethnicity | Economic | LEP | SCI |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | M | Lewis | 4.0 | 43954 | 1.0 | 0.0 | 0.0 | 0.0 | |
| **1** | 1 | F | Howell | 6.0 | 47436 | 0.0 | 0.0 | 0.0 | 0.0 | |
| **2** | 2 | m | Lewis | 4.0 | 59755 | 0.0 | 0.0 | 0.0 | 0.0 | |
| **3** | 3 | M | Marshall | 5.0 | 35449 | 1.0 | 1.0 | 0.0 | 0.0 | |
| **4** | 4 | M | Lewis | 3.0 | 43956 | 1.0 | 1.0 | 0.0 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **395** | 433 | F | Frederiksen | 6.0 | 56703 | 0.0 | 3.0 | 0.0 | 1.0 | |
| **396** | 434 | M | Lehmann | 2.0 | 57499 | 1.0 | 3.0 | 1.0 | 1.0 | |
| **397** | 435 | F | Brennan | 2.0 | 65507 | 0.0 | 3.0 | 1.0 | 1.0 | |
| **398** | 436 | M | Howell | 4.0 | 36145 | 1.0 | 5.0 | 1.0 | 1.0 | |
| **399** | 437 | M | Howell | 4.0 | 39634 | 1.0 | 5.0 | 1.0 | 1.0 | |

399 rows × 33 columns

```
df = df.dropna() # drop null values
#consider using mean as well
```

```
df.isnull()
```

| | Gender | Teacher | Period | Student ID | Gender.1 | Ethnicity | Economic | LEP | SCICOUR | ( |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | False | False | False | False | Fa |
| **1** | False | False | False | False | False | False | False | False | False | Fa |
| **2** | False | False | False | False | False | False | False | False | False | Fa |
| **3** | False | False | False | False | False | False | False | False | False | Fa |
| **4** | False | False | False | False | False | False | False | False | False | Fa |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |

```
df['Gender.1']=df['Gender.1'].astype('float64') #these were type object for some reason
df['Ethnicity']=df['Ethnicity'].astype('float64')
df['SCICOUR']=df['SCICOUR'].astype('float64')
```

| **456** | False | False | False | False | False | False | False | False | False | Fa |
|---|---|---|---|---|---|---|---|---|---|---|

```
r,c = df2.shape #by only dropping TOSLSTOT columns and imputing the mean, we increased our da
r,c
```

```
(399, 33)
```

```
r,c = df.shape
r,c
```

```
(360, 32)
```

After removing null value, we have 360 rows and 32 columns

```
df.drop_duplicates() #checking duplicate
```

| | Gender | Teacher | Period | Student ID | Gender.1 | Ethnicity | Economic | LEP | SCICOUR | CTE |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | M | Lewis | 4.0 | 43954 | 1.0 | 0.0 | 0.0 | 0.0 | 1.5 | 0.0 |

No duplicate data

```
df.dtypes #need to fix datatypes
```

```
Gender          object
Teacher         object
Period         float64
Student ID      object
Gender.1       float64
Ethnicity      float64
Economic       float64
LEP            float64
SCICOUR        float64
CTE            float64
TOT             object
BIO            float64
ELA            float64
Alg            float64
GPA            float64
TOSLS1         float64
TOSLS2         float64
TOSLS3         float64
TOSLS4         float64
TOSLS5         float64
TOSLS6         float64
TOSLS7         float64
TOSLS8         float64
TOSLS9         float64
TOSLSTOT       float64
BRAINS1        float64
BRAINS2        float64
BRAINS3        float64
BRAINS4        float64
BRAINS5        float64
BRAINSTOT      float64
SORT           float64
dtype: object
```

```
#df.replace("m","M") # replace m with M
```

```
#df = df[df.Gender != "1- 4"]
```

```
#r,c = df.shape
#r,c
```

```
#df
```

# Exploratory Data Analysis

```
"""Data Dictionary - we made this to help us with our interpretation
this dictionary created by us from reading the research papers

Gender          M or F
Teacher         categorical grouping of students
Period         numeric category for grouping students by location
Student ID      identifier for student
Gender.1        1=M, 0=F
Ethnicity       0=American Indian, 1=Asian, 2=Black, 3=Hispanic 4=Two or More, 5=White ** Do p
Economic       1=in economic need, 0=not in economic need (defined by free and reduced lunch p
LEP            0 = limited english proficiency, 1=proficient in english
SCICOUR         number of  course credits earned in science classes
CTE            number of course credits earned in career/tech/engineering classes
TOT             ?
BIO            grade on biology state assessment
ELA            grade on english state assessment
Alg            grade on algebra state assessment
GPA            grade point average in high school
TOSLS1         these are measurements of scientific literacy
TOSLS2         float64
TOSLS3         float64
TOSLS4         float64
TOSLS5         float64
TOSLS6         float64
TOSLS7         float64
TOSLS8         float64
TOSLS9         float64
TOSLSTOT       I think this is total of TOSLS 1 through 9?
BRAINS1        float64
BRAINS2        float64
BRAINS3        float64
BRAINS4        float64
BRAINS5        float64
BRAINSTOT      "Behavior, related attitudes, and intentions towards science" survey info
SORT           float64
"""
```

```
'Data Dictionary - we made this to help us with our interpretation\nthis dictionary crea
categorical grouping of students\nPeriod        numeric category for grouping students b
Ethnicity      0=American Indian, 1=Asian, 2=Black, 3=Hispanic 4=Two or More, 5=White **
ned by free and reduced lunch program)\nLEP           0 = limited english proficiency, 1
cience classes\nCTE           number of course credits earned in career/tech/engineering
\nELA          grade on english state assessment\nAlg          grade on algebra state
```

```
df.describe()
```

| | Period | Gender.1 | Ethnicity | Economic | LEP | SCICOUR | CTE |
|---|---|---|---|---|---|---|---|
| **count** | 360.000000 | 360.000000 | 360.00000 | 360.000000 | 360.000000 | 360.000000 | 360.000000 |
| **mean** | 4.394444 | 0.425000 | 4.22500 | 0.211111 | 0.066667 | 2.458333 | 0.797222 |
| **std** | 2.050877 | 0.495031 | 1.23186 | 0.408665 | 0.249791 | 0.789688 | 1.066973 |
| **min** | 1.000000 | 0.000000 | 0.00000 | 0.000000 | 0.000000 | 0.500000 | 0.000000 |
| **25%** | 3.000000 | 0.000000 | 3.00000 | 0.000000 | 0.000000 | 1.500000 | 0.000000 |
| **50%** | 4.000000 | 0.000000 | 5.00000 | 0.000000 | 0.000000 | 2.500000 | 0.500000 |
| **75%** | 6.000000 | 1.000000 | 5.00000 | 0.000000 | 0.000000 | 3.000000 | 1.000000 |
| **max** | 8.000000 | 1.000000 | 5.00000 | 1.000000 | 1.000000 | 4.500000 | 5.000000 |

8 rows × 28 columns

```
df['Gender'].value_counts() #looks like we have 8 values we can clean up for increment 2
```

```
F        204
M        148
1 - 4      7
m          1
Name: Gender, dtype: int64
```

```
df['Ethnicity'].value_counts()
```

```
5.0    240
3.0     64
2.0     19
4.0     19
1.0     15
0.0      3
Name: Ethnicity, dtype: int64
```

```
df['LEP'].value_counts()
```

```
0.0    336
1.0     24
Name: LEP, dtype: int64
```

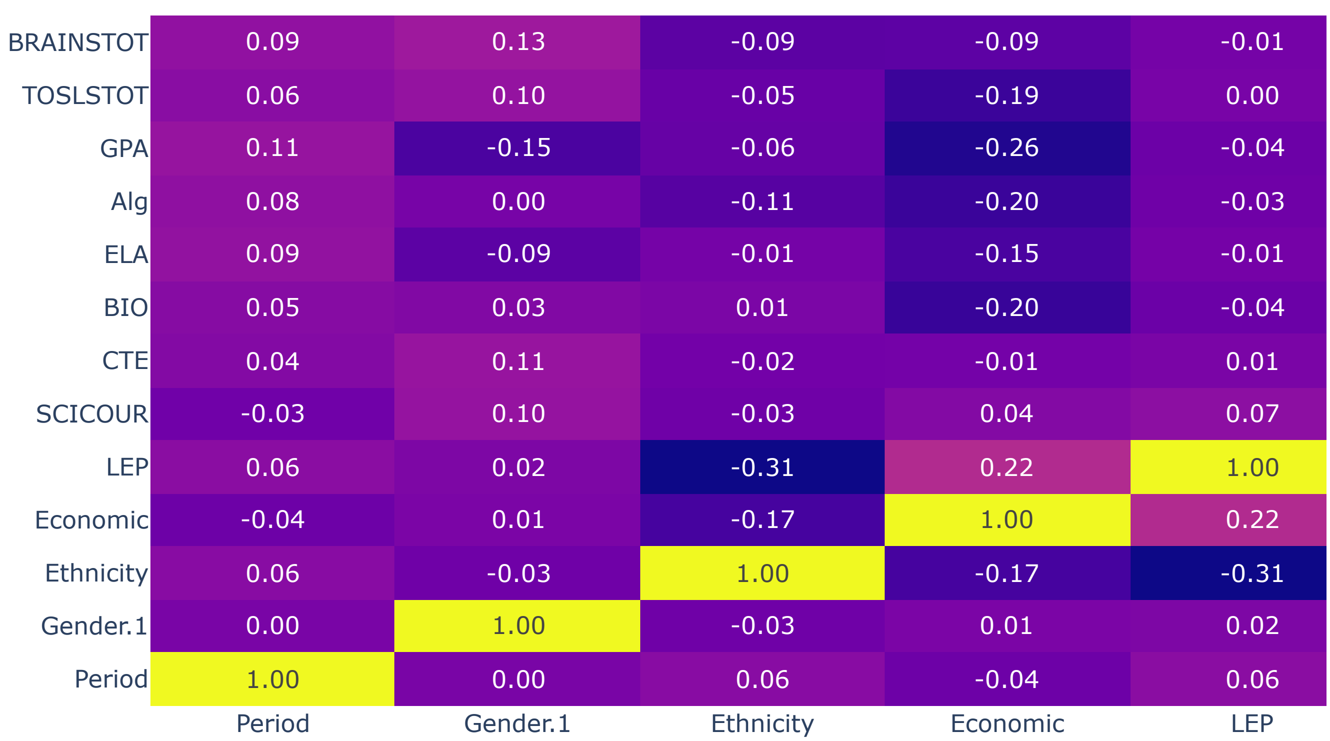
```
df['Economic'].value_counts()
```

```
0.0    284
1.0     76
Name: Economic, dtype: int64
```

```
import plotly
import plotly.graph_objects as go


df_cor=df.drop(columns=['TOSLS1', 'TOSLS2', 'TOSLS3', 'TOSLS4', 'TOSLS5', 'TOSLS6', 'TOSLS7',
correlation=df_cor.corr()


fig = go.Figure()
fig.add_trace(
    go.Heatmap(
        x = correlation.columns,
        y = correlation.index,
        z = np.array(correlation),
        text=correlation.values,
        texttemplate='%{text:.2f}'
    )
)
fig.show()
```

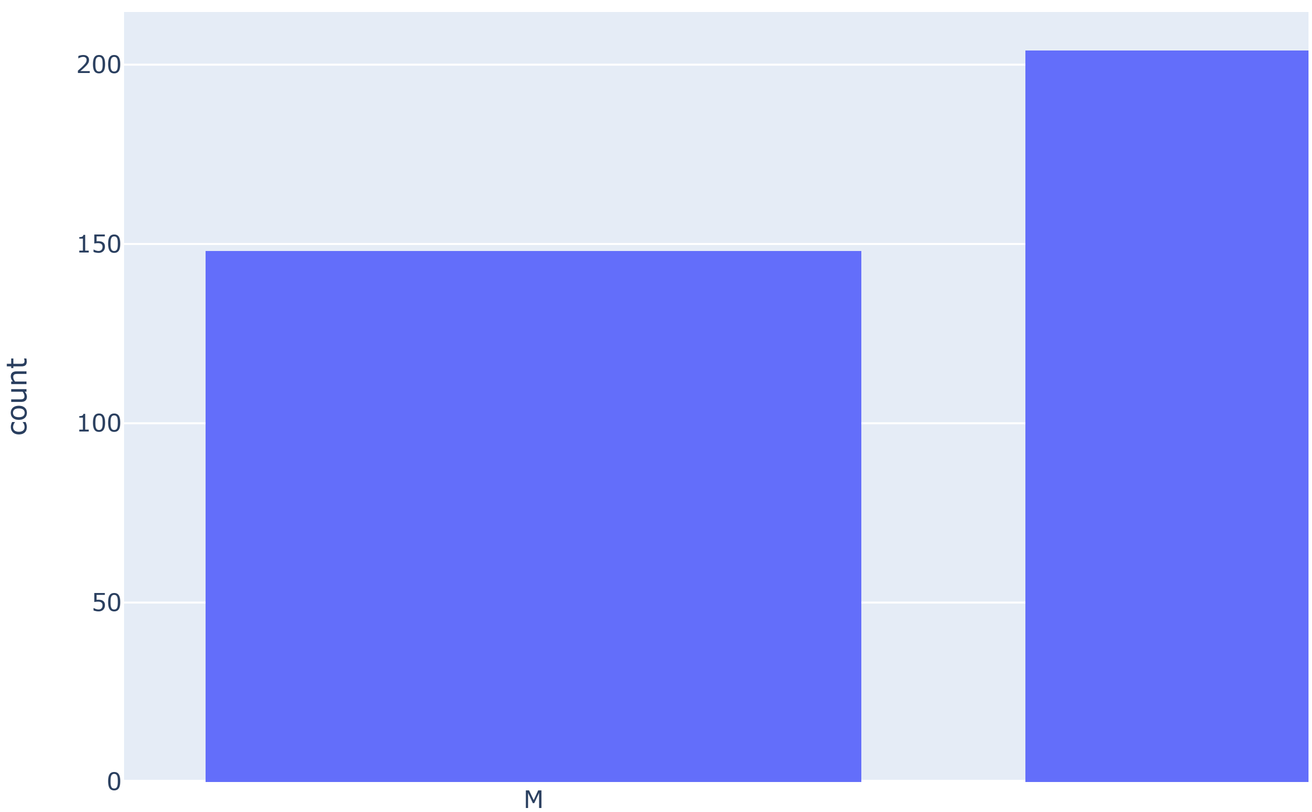



```
"""some interesting stuff in this correlation heatmap
```

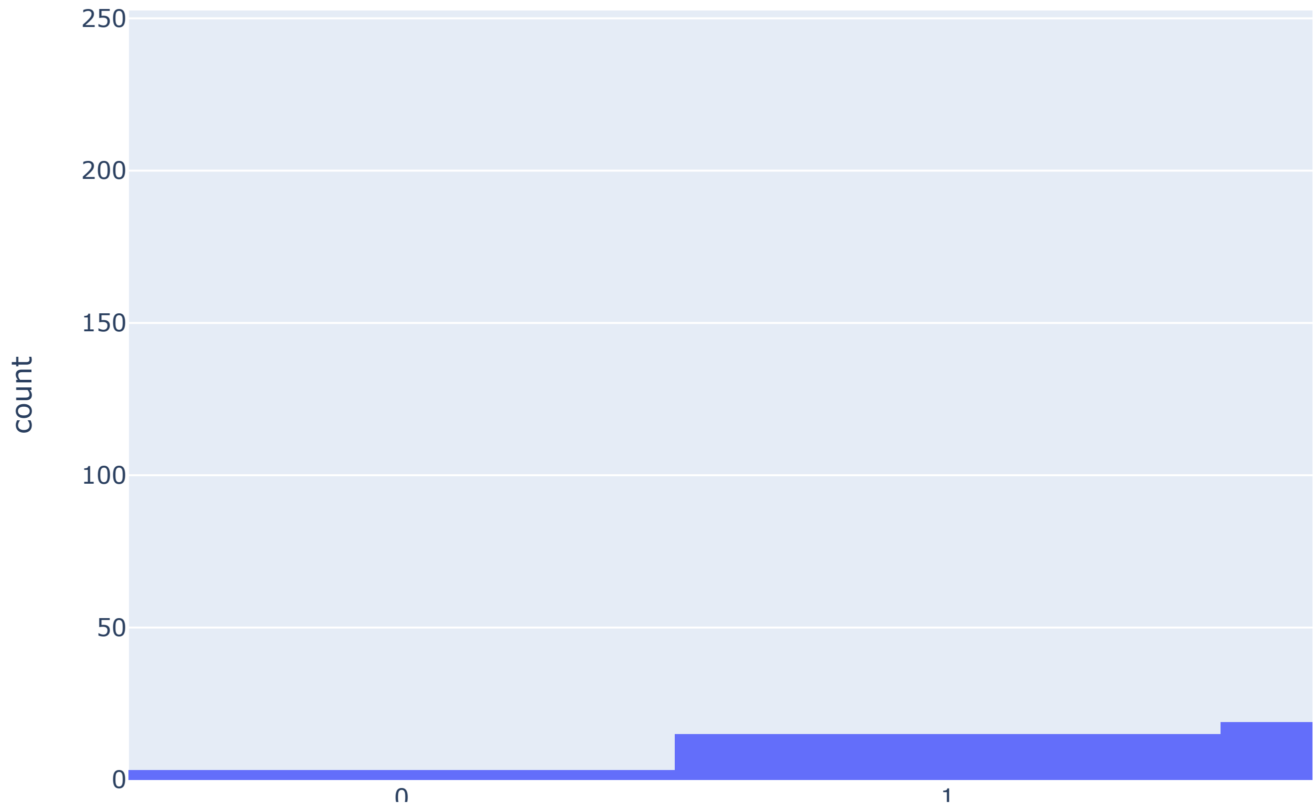
```
highest correlation is ELA and BIO which is the English and Science tests. you might think Ma
Unsurprising, the 3 standardized tests (BIO, ELA, ALG) and GPA are much more correlated than
its sad socially that LEP and Economic are somewhat correlated,, but nice that LEP and GPA ar
"""
```

```
'some interesting stuff in this correlation heatmap\nhighest correlation is ELA and BIO
ld be higher\nUnsurprising, the 3 standardized tests (BIO, ELA, ALG) and GPA are much mo
somewhat correlated,, but nice that LEP and GPA are not correlated\n'
```

```
fig = px.histogram(df, x="Gender")
fig.show()
#looks like Gender needs cleaning. maybe it the Gender.1 is cleaned already?
```

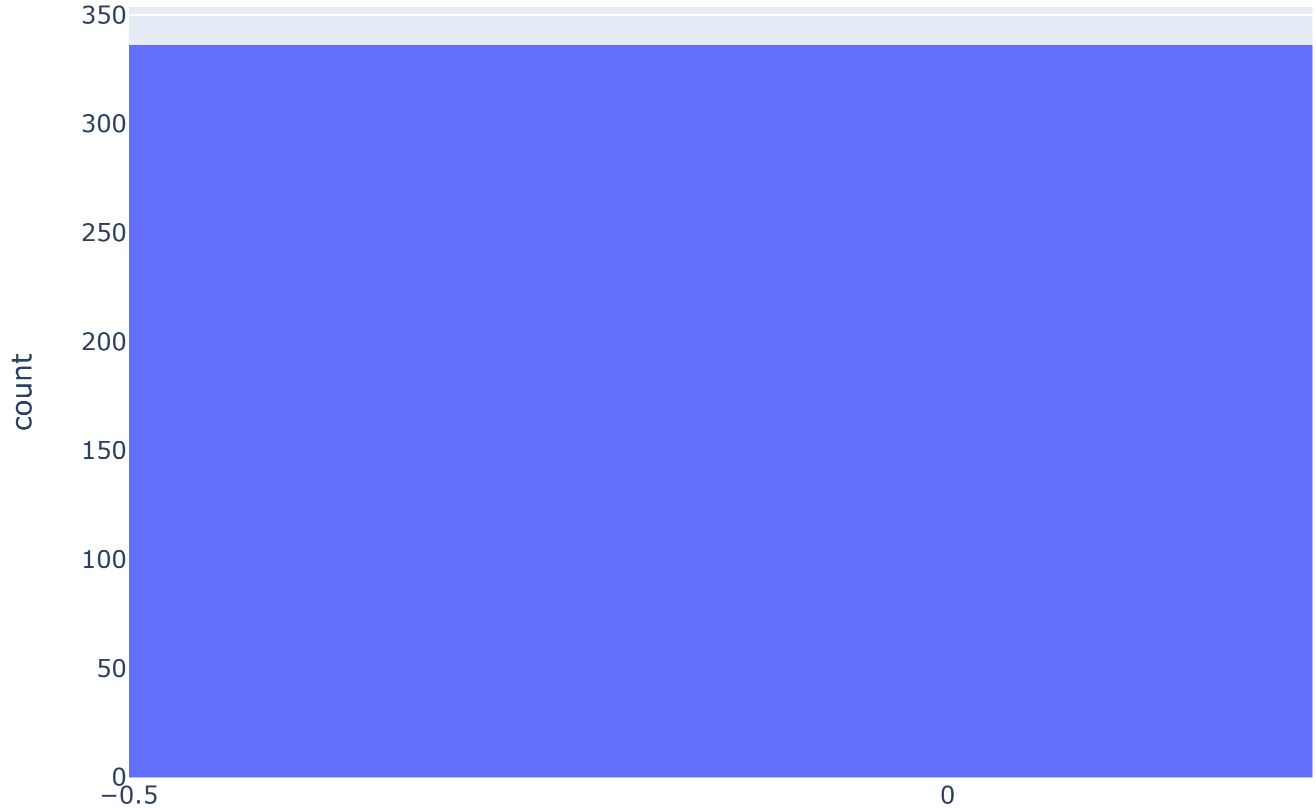



```
fig = px.histogram(df, x="Ethnicity")
fig.show()
#need to update the labels here from the data dictionary #done
#some class imbalance
```

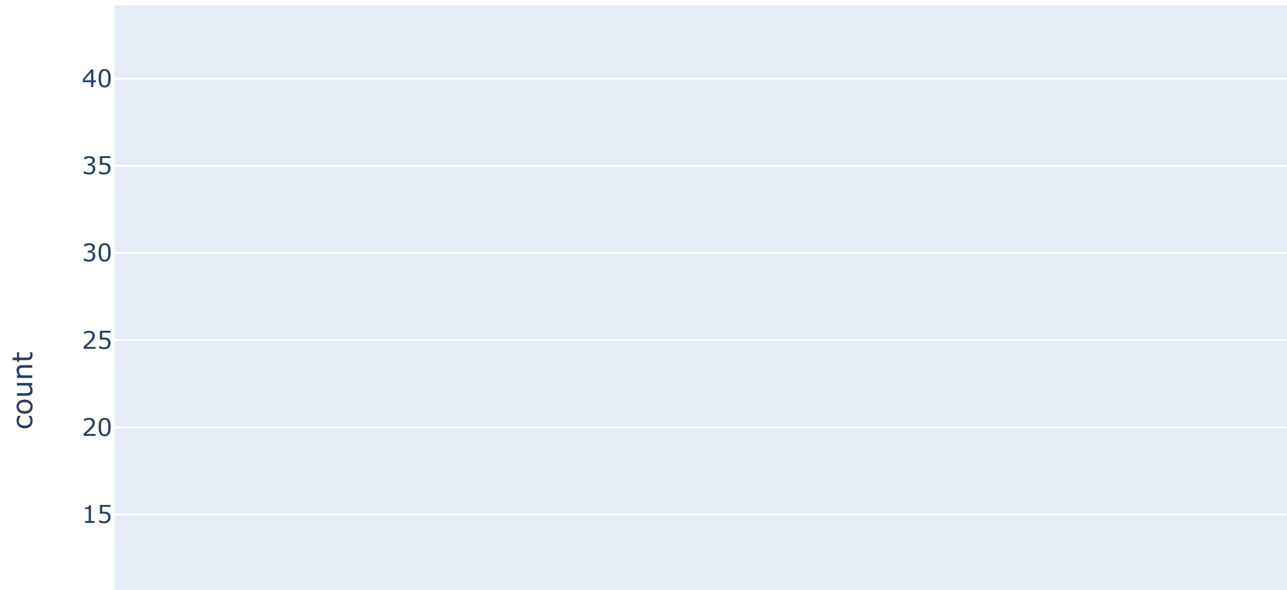



```
fig = px.histogram(df, x="Economic")
fig.show()
```

```
fig = px.histogram(df, x="LEP")
fig.show()
```

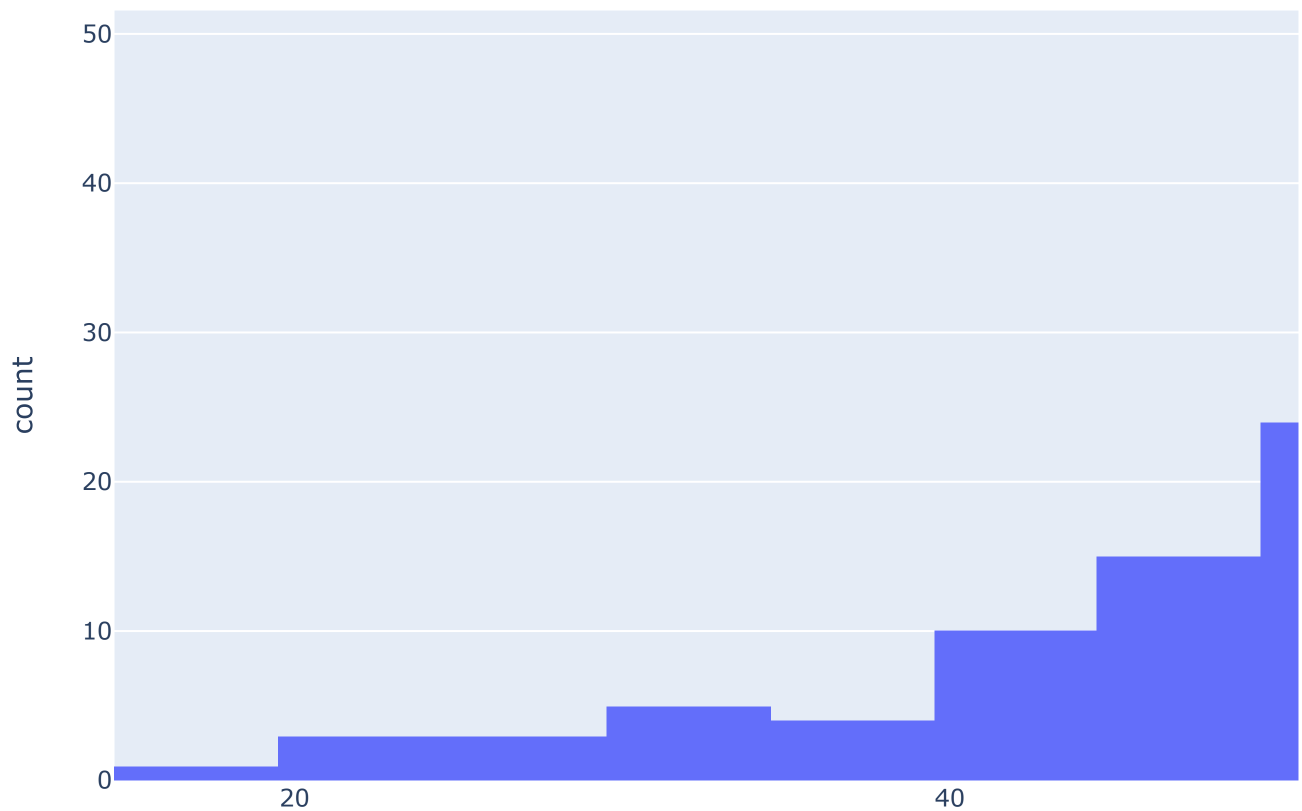



```
fig = px.histogram(df, x="GPA")
fig.show()
#not a normal distribution
```

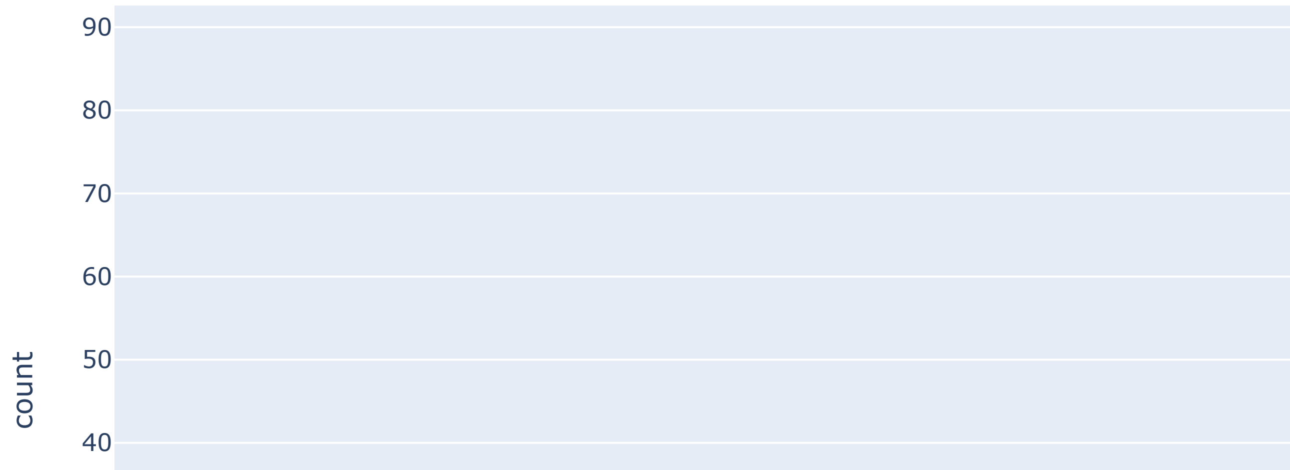



```
fig = px.histogram(df, x="TOSLSTOT")
fig.show()
#this looks fairly normal, might play with bin size
```

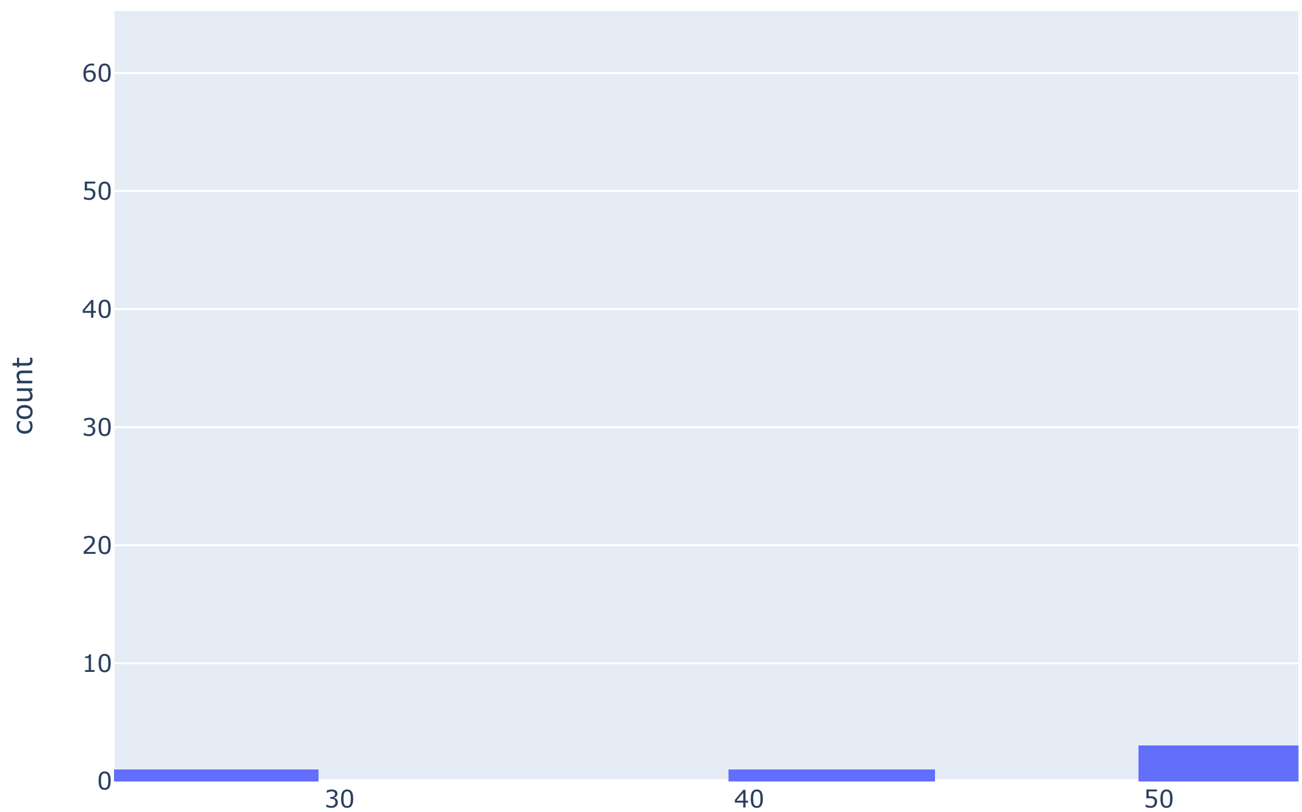
```
fig = px.histogram(df, x="BRAINSTOT")
fig.show()
#interesting similar to TOSLSTOT
```




```
fig = px.histogram(df, x="BIO")
fig.show()
#strong tail
```

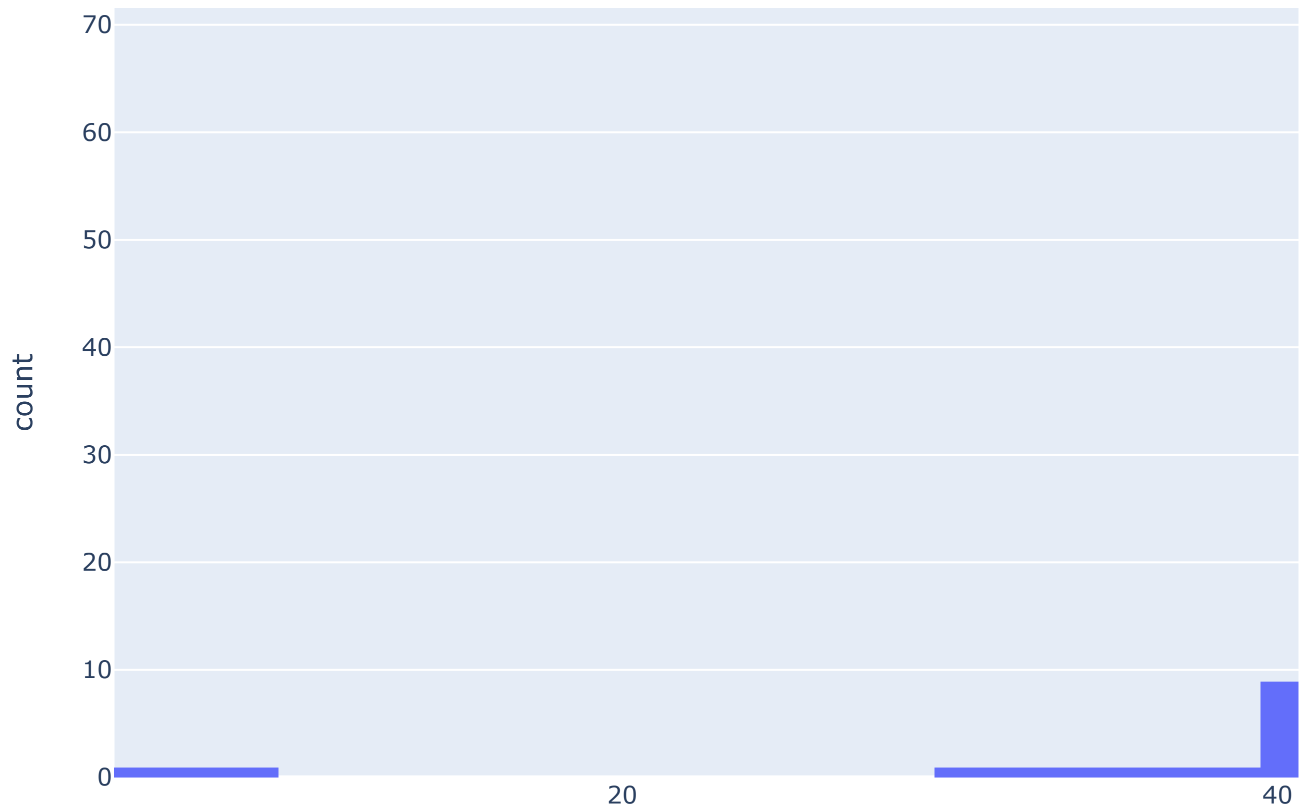



```
fig = px.histogram(df, x="ELA")
fig.show()
#mostly normal with some skew
```

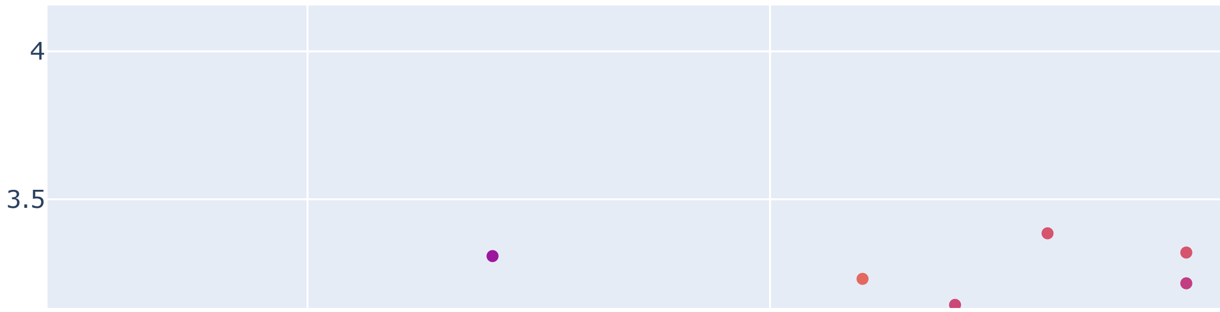



```
fig = px.histogram(df, x="Alg")
```

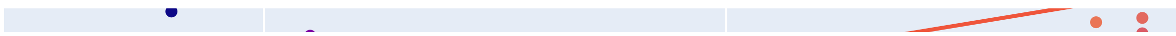
```
fig.show()
```




```
fig = px.scatter(x=df['BIO'], y=df['GPA'], color=df['ELA'], trendline="ols")
fig.show()
#can see the general pattern between BIO GPA, indicitive that a linear regression would do...
```




```
#pie chart showing distribution of categorical variables (gender, ethnicity, lep, economic)
#histogram showing frequency distribution of continuous (gpa, toslstot, brainstot, bio, ela,
```



```
fig = px.pie(df, values='GPA', names='Ethnicity')
fig.show()
```

```
fig = px.pie(df, values='ELA', names='Ethnicity')
fig.show()
```

```
fig = px.pie(df, values='LEP', names='Ethnicity')
fig.show()
```

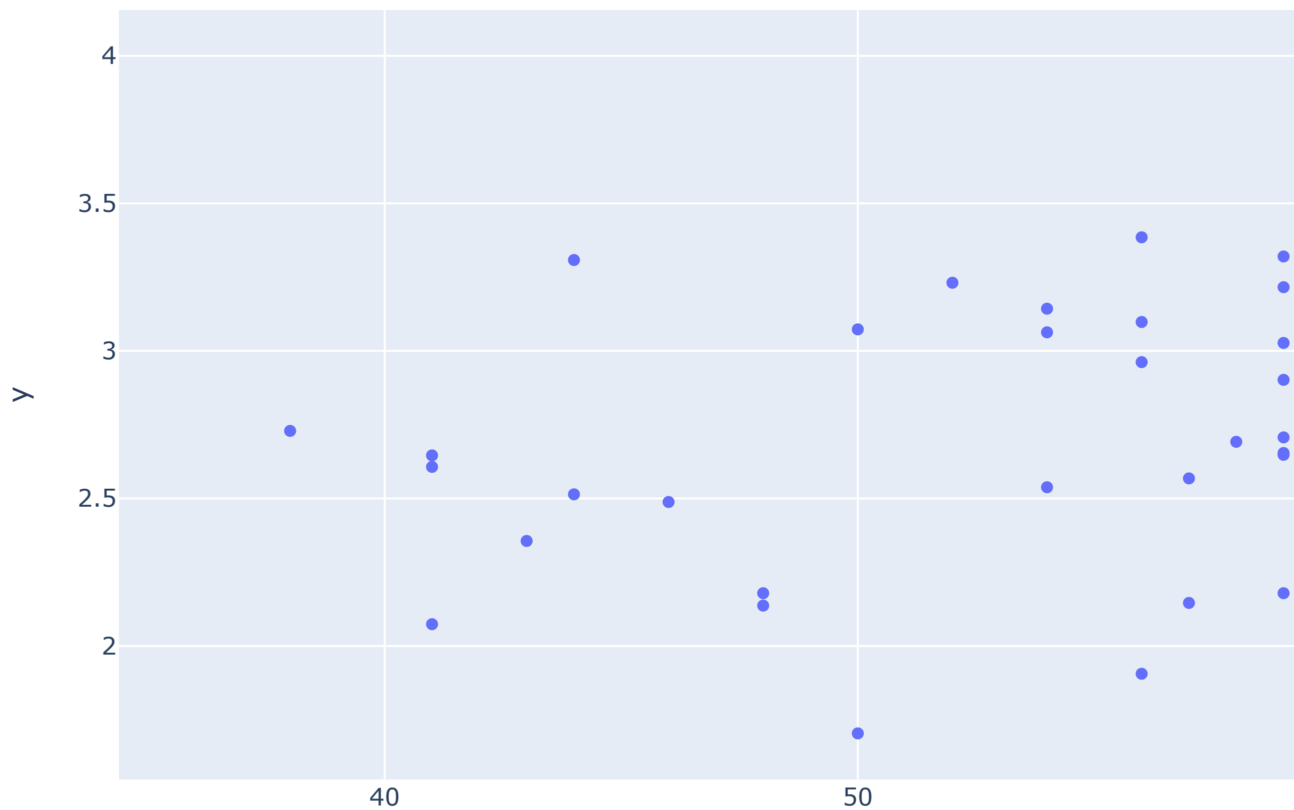
```
fig = px.pie(df, values='Economic', names='Ethnicity')
fig.show()
```

```
fig = px.pie(df, values='LEP', names='Gender')
fig.show()
```

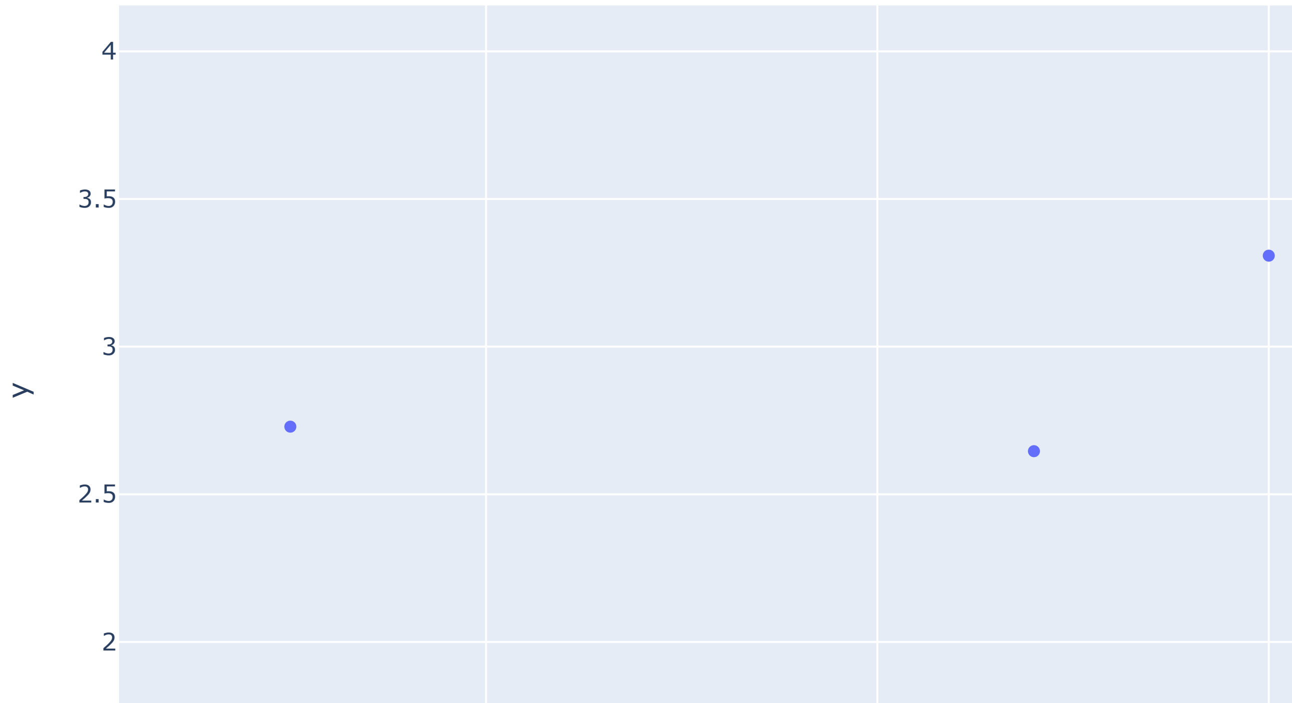
```
fig = px.pie(df, values='Economic', names='Gender')
fig.show()
```

```
#scatter of BIO vs GPA, ELA vs GPA, Alg vs GPA
```

```
# scatter of BIO vs GPA
fig = px.scatter(x=df['BIO'], y=df['GPA'])
fig.show()
#not as good as our plotly, but matplotlib does get smae results
```



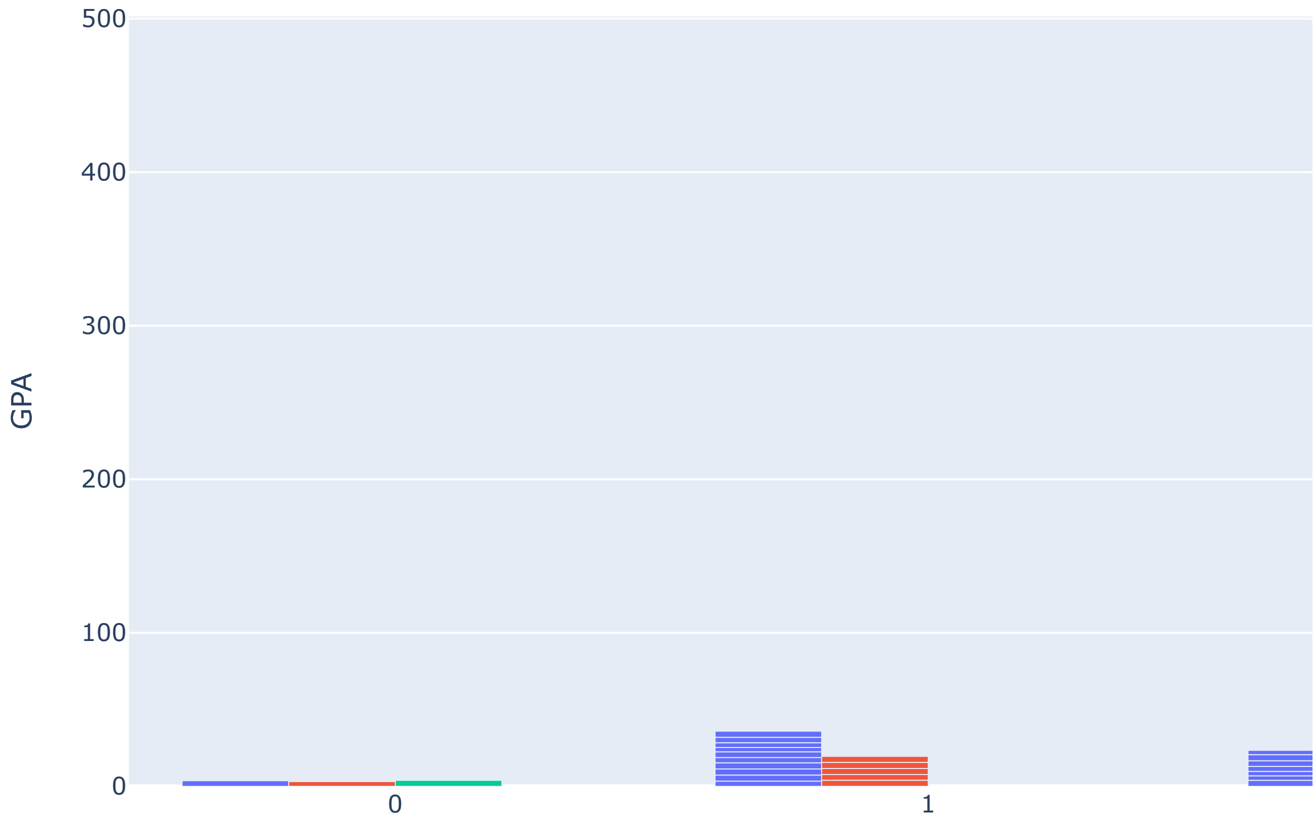

```
# scatter of ELA vs GPA
fig = px.scatter(x=df['ELA'], y=df['GPA'])
fig.show()
#tighter grouping than BIO
```




```
# scatter of Alg vs GPA
fig = px.scatter(x=df['Alg'], y=df['GPA'])
fig.show()
```

```
# Ethic - GPA/Economic with bar chart groupby()
# Gender - Bio/ Engl/ Algebra
```

```
fig = px.bar(df, x="Ethnicity", y="GPA", color='Gender',
              barmode='group')
fig.show()
```




```
#prove the assumptions of t-test and anova (assumption of normality, assumption of homogeneit
#then run some t-tests and some anovas
#anova in ICE4
```

# Regression

```
#use a scaler? normalization? regularization? this might be done in increment2
#original paper used multiple linear regression, we can test other models. decision tree regr
```

```
df_dummies=pd.get_dummies(df['Ethnicity']) #one hot encoding the ethnicity variables
df_dummies.columns=['AI', 'AN', 'BK', 'HC', 'TE', 'WE'] #renaming them to match the data dict
df_onehot=df.join(df_dummies) #joining the one hot dataframe with the other features
```

```
df_onehot
```

| | Gender | Teacher | Period | Student ID | Gender.1 | Ethnicity | Economic | LEP | SCICOUR |
|---|---|---|---|---|---|---|---|---|---|
| **0** | M | Lewis | 4.0 | 43954 | 1.0 | 0.0 | 0.0 | 0.0 | 1.5 |
| **1** | F | Howell | 6.0 | 47436 | 0.0 | 0.0 | 0.0 | 0.0 | 3.5 |
| **2** | m | Lewis | 4.0 | 59755 | 0.0 | 0.0 | 0.0 | 0.0 | 1.5 |
| **3** | M | Marshall | 5.0 | 35449 | 1.0 | 1.0 | 0.0 | 0.0 | 3.5 |
| **4** | M | Lewis | 3.0 | 43956 | 1.0 | 1.0 | 0.0 | 0.0 | 1.5 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **434** | M | Lehmann | 2.0 | 57499 | 1.0 | 3.0 | 1.0 | 1.0 | 2.0 |
| **435** | F | Brennan | 2.0 | 65507 | 0.0 | 3.0 | 1.0 | 1.0 | 3.5 |
| **436** | M | Howell | 4.0 | 36145 | 1.0 | 5.0 | 1.0 | 1.0 | 3.5 |
| **437** | M | Howell | 4.0 | 39634 | 1.0 | 5.0 | 1.0 | 1.0 | 2.5 |
| **438** | F | Nyholm | 5.0 | 40738 | 0.0 | 5.0 | 1.0 | 1.0 | 2.5 |

360 rows × 38 columns

```
from sklearn.model_selection import cross_val_score
from numpy import absolute
```

```
X=df_onehot.drop(columns=['TOSLSTOT', 'TOSLS1', 'TOSLS2', 'TOSLS3', 'TOSLS4', 'TOSLS5', 'TOSL
y=df['TOSLSTOT']
#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
regressor = xgb.XGBRegressor(n_estimators=100,reg_lambda=1,gamma=0,max_depth=3,objective ='re
RMSE_list=[]
kf = KFold(n_splits=10, random_state=42, shuffle=True)
RMSE_list = cross_val_score(regressor, X, y, scoring='neg_mean_absolute_error',cv=kf, n_jobs=
print("The RMSE is " + str(mean(absolute(RMSE_list))))
```

```
The RMSE is 3.1729476504855687
```

```
#this is the model from the paper from the multilinear regression done by Chandler et al. quo
```

```
X_test_paper=df_onehot.drop(columns=['CTE', 'TOSLSTOT', 'TOSLS1', 'TOSLS2', 'TOSLS3', 'TOSLS4
y_test=df_onehot['TOSLSTOT']
chandler_weights=[.807,-1.049,.936,1.772,.087,.119,.020,.668,.041,2.511,1.231,.858,-0.893,1.0
y_test_paper_preds=X_test_paper @ chandler_weights + -14.212
RMSE = np.sqrt(mean_squared_error(y_test, y_test_paper_preds))
print("The RMSE is " + str(RMSE))
```

```
The RMSE is 3.713590332197263
```

```
#model from the paper full equation with mapping to our dataframe
#the weights matrix above is extracted from this equation
"""
TOSLS = -14.212 – 1.049(EN) + .936(EP) + .668(GP) + 1.772(SC) + .087(SB) + .041(AS) +
.119(SE) + .020(SA) + .807(GE) + 2.511(AI) + 1.231(AN) + .858(BK) – .893(HC) + 1.065(TE)
+ .234(WE)
EN = 0, 1 GP = 0.0 – 4.0 SC = 0 – (4.5) AS = 0 - 114
EP = 0, 1 SB = 0 – 100 SE = 0 – 100 GE = 0, 1
AI, AN, BK, HC, TE, WE = 0, 1 TOSLS = 0 – 28

MAPPING ABBREVIATIONS TO DATAFRAME HEADER
 (Constant)
EN=Economic
EP=LEP
GP=GPA
SC=SCICOUR
BRAINS TOTAL (AS)=BRAINSTOT
STAAR Algebra (SA) = ALG
STAAR Biology Percentage (SB) = BIO
STAAR English (SE) = ENG
Ethnicity (AI, AN, BK, HC, TE, WE) 0=American Indian, 1=Asian, 2=Black, 3=Hispanic 4=Two or M
Gender = Gender.1

"""
```

```
'\nTOSLS = -14.212 – 1.049(EN) + .936(EP) + .668(GP) + 1.772(SC) + .087(SB) + .041(AS) +
C) + 1.065(TE)\n+ .234(WE)\nEN = 0, 1 GP = 0.0 – 4.0 SC = 0 – (4.5) AS = 0 - 114\nEP = 0
= 0 – 28\n\nMAPPING ABBREVIATIONS TO DATAFRAME HEADER\n (Constant)\nEN=Economic\nEP=LEP\
STAAR Biology Percentage (SB) = BIO\nSTAAR English (SE) = ENG\nEthnicity (AI, AN, BK, HC
hite= AI, AN, BK, HC, TE, WE\nGender = Gender.1\n\n'
```

```
#this model is using 399 rows with 39 rows having imputed means
#by adding those 39 rows in we improve our RMSE by .02
df2_dummies=pd.get_dummies(df2['Ethnicity']) #one hot encoding the ethnicity variables
df2_dummies.columns=['AI', 'AN', 'BK', 'HC', 'TE', 'WE'] #renaming them to match the data dic
df2_onehot=df2.join(df2_dummies) #joining the one hot dataframe with the other features
X2=df2_onehot.drop(columns=['TOSLSTOT', 'TOSLS1', 'TOSLS2', 'TOSLS3', 'TOSLS4', 'TOSLS5', 'TO
y2=df2['TOSLSTOT']
RMSE_list=[]
kf = KFold(n_splits=10, random_state=42, shuffle=True)
RMSE_list = cross_val_score(regressor, X2, y2, scoring='neg_mean_absolute_error',cv=kf, n_job
print("The RMSE is " + str(mean(absolute(RMSE_list))))
```

```
The RMSE is 3.150916702105449
```

```
#using Support Vector regression as a comparison to our XGBoost Regressor. XGB won out, but S
from sklearn.svm import SVR

regressor_svm = SVR(kernel = 'rbf')
RMSE_list=[]
kf = KFold(n_splits=10, random_state=42, shuffle=True)
RMSE_list = cross_val_score(regressor_svm, X, y, scoring='neg_mean_absolute_error',cv=kf, n_j
print("The RMSE is " + str(mean(absolute(RMSE_list))))
```

```
The RMSE is 3.342278549444955
```

```
from sklearn.feature_selection import SelectFromModel  #these is an attempt to see if reducin
from sklearn.ensemble import RandomForestClassifier
sel = SelectFromModel(RandomForestClassifier(n_estimators = 50))
X=df_onehot.drop(columns=['TOSLSTOT', 'TOSLS1', 'TOSLS2', 'TOSLS3', 'TOSLS4', 'TOSLS5', 'TOSL
y=df['TOSLSTOT']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
sel.fit(X_train, y_train)
print("The best features to use are: ")
feature_index = sel.get_support(indices=True)
X_new=X.iloc[:,feature_index]
print(X_new.head())
RMSE_list = cross_val_score(regressor, X_new, y, scoring='neg_mean_absolute_error',cv=kf, n_j
print("The RMSE for reduced feature list with XGB is " + str(mean(absolute(RMSE_list))))
RMSE_list = cross_val_score(regressor_svm, X_new, y, scoring='neg_mean_absolute_error',cv=kf,
print("The RMSE for reduced feature list with SVM is " + str(mean(absolute(RMSE_list))))
```

```
The best features to use are:
   SCICOUR  CTE   BIO   ELA   Alg    GPA  BRAINSTOT
0      1.5  0.0  88.0  91.0  98.0  3.654       79.0
1      3.5  0.0  63.0  60.0  65.0  3.094       59.0
2      1.5  1.5  94.0  93.0  98.0  3.893       77.0
3      3.5  1.5  67.0  69.0  91.0  3.392       90.0
4      1.5  0.5  90.0  91.0  93.0  3.923       87.0
The RMSE for reduced feature list with XGB is 3.210232792960273
The RMSE for reduced feature list with SVM is 3.2884826129963947
```

```
weights=sel.estimator_.feature_importances_
weights_percent = 100 * (weights/max(weights))
for i in range(0,len(weights)):
  print("The feature " + str(X.columns[i]) + " has a relative importance percentage " + str(w
```

```
The feature Gender.1 has a relative importance percentage 24.217217000833756
The feature Economic has a relative importance percentage 19.1831786069307
The feature LEP has a relative importance percentage 6.315086350374305
The feature SCICOUR has a relative importance percentage 51.94848112742332
The feature CTE has a relative importance percentage 54.089276486354485
```

```
The feature BIO has a relative importance percentage 96.95978301326308
The feature ELA has a relative importance percentage 89.14472498769913
The feature Alg has a relative importance percentage 85.28409765179161
The feature GPA has a relative importance percentage 100.0
The feature BRAINSTOT has a relative importance percentage 98.82942039205139
The feature AI has a relative importance percentage 2.9460421746837713
The feature AN has a relative importance percentage 6.452192623812423
The feature BK has a relative importance percentage 6.34040472862085
The feature HC has a relative importance percentage 13.78984692352251
The feature TE has a relative importance percentage 7.827270372687159
The feature WE has a relative importance percentage 17.440363964560625
```