COGS 109 Project Notebook

SAT Scores: A Culmination of Contributing Factors or an Isolated Score?

Authors: Alexander Schonken, Jolene Leung, Bingzhe Wang

In [1]:
```
from mpl_toolkits import mplot3d
import pandas as pd
pd.options.mode.chained_assignment = None  # default='warn'
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import math


corgis_df = pd.read_csv("./school_scores.csv")
corgis_df.head()
```

Out[1]:

	Year	State.Code	State.Name	Total.Math	Total.Test-takers	Total.Verbal	Academic Subjects.Arts/Music.Average GPA	Subj
0	2005	AL	Alabama	559	3985	567	3.92	
1	2005	AK	Alaska	519	3996	523	3.76	
2	2005	AZ	Arizona	530	18184	526	3.85	
3	2005	AR	Arkansas	552	1600	563	3.90	
4	2005	CA	California	522	186552	504	3.76	

5 rows × 99 columns

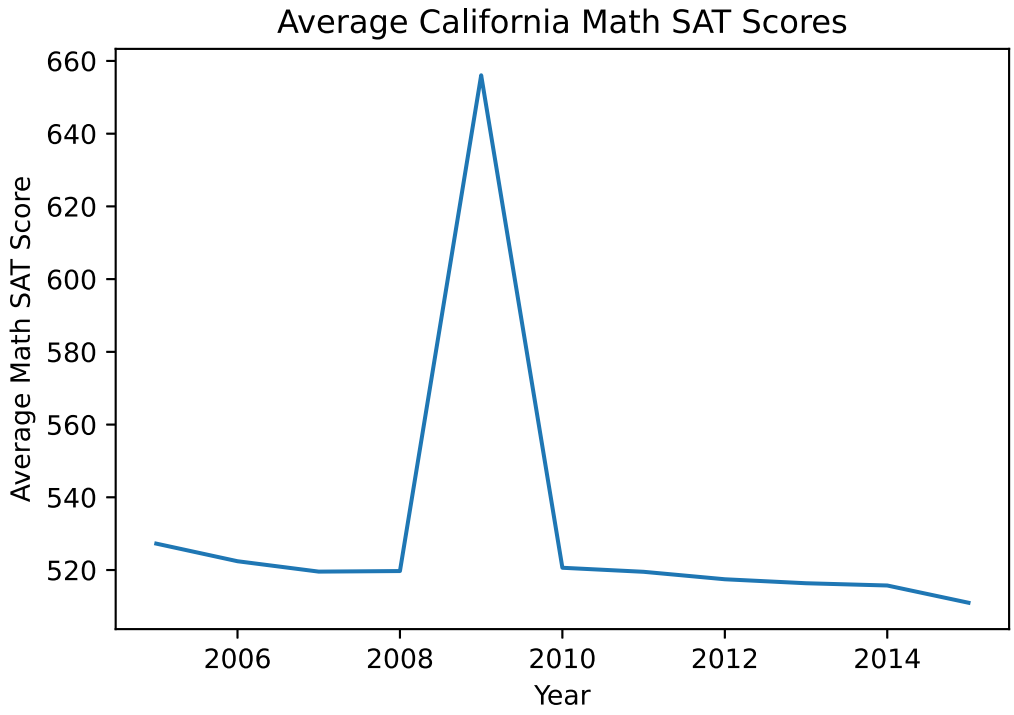
Background

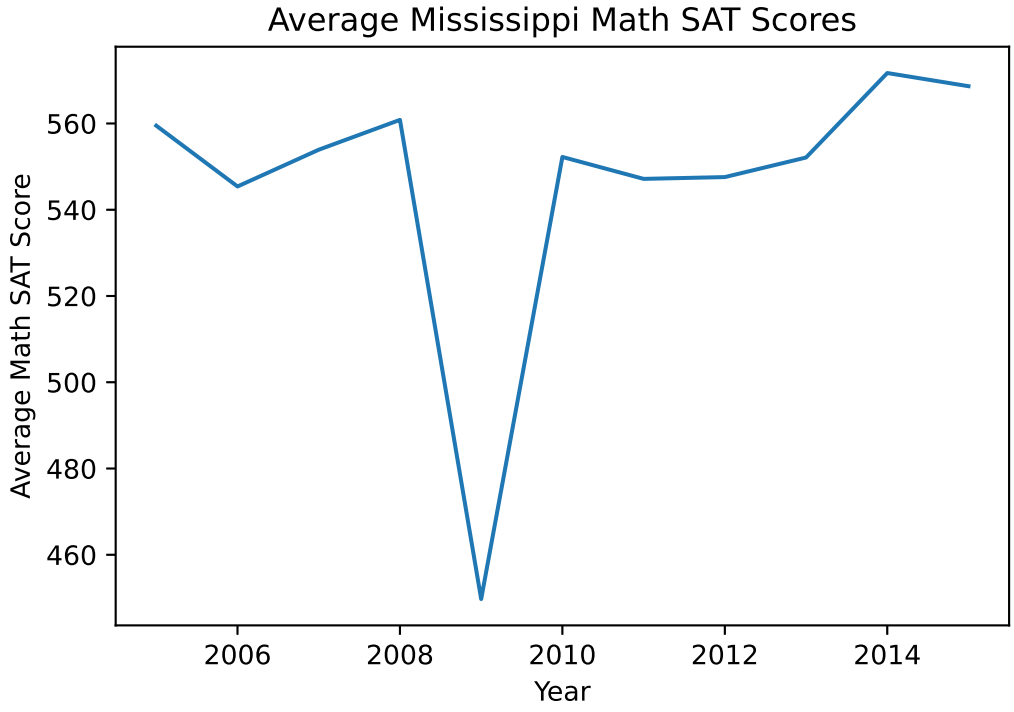
The School Scores Dataset from the CORGIS Dataset Project is a set of 577 samples and 99 features. Each sample is attached to a specific state in the United States of America and a year (2005-2015). The features range in different school related student attributes ranging from the average Math score of students to the number of students with yearly family income greater than $100k. The dataset does deal with data over time but we will carefully work around that as we progress from analysis to model. The dataset can be downloaded at the following link: CORGIS School Scores CSV File

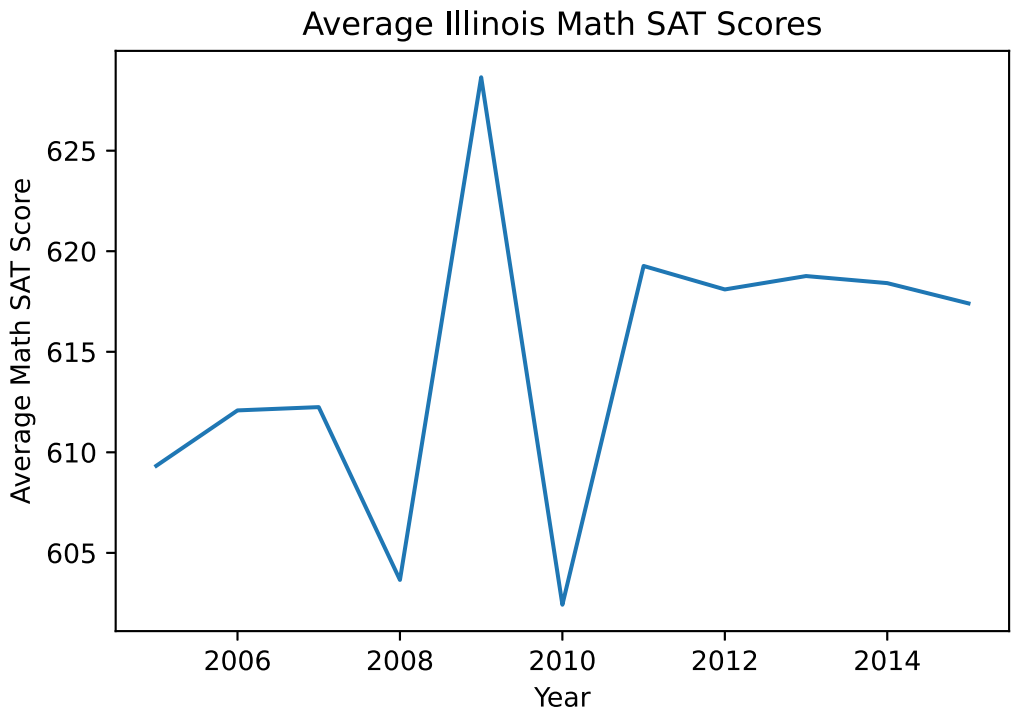
Before we begin out attempt at training a linear regression model, we will analyse the data across states, time, income, and other features to look for correlation and an interesting avenue by which to discover new insights with our model.

In [2]:

```
# Analysis 1: California, Mississippi, and Illinois SAT scores between 600-800 math tot
# Reason: Pick 2 different states across the country from one another to judge of effec

# Pull out California and Mississippi data
cal_df = corgis_df[corgis_df["State.Name"] == "California"]
miss_df = corgis_df[corgis_df["State.Name"] == "Mississippi"]
ill_df = corgis_df[corgis_df["State.Name"] == "Illinois"]

# Plot year vs SAT score total students 600-800 math
# California
tot_stu_df = cal_df[["Score Ranges.Between 200 to 300.Math.Total", "Score Ranges.Betwee
tot_stu_score_df = cal_df[["Score Ranges.Between 200 to 300.Math.Total", "Score Ranges.
for i, name in enumerate(tot_stu_score_df.columns):
    tot_stu_score_df[name] = tot_stu_score_df[name] * (250 + (i * 100))
tot_stu_averages = tot_stu_score_df.reset_index(drop="true").sum(axis=1) / tot_stu_df.r

plt.plot(cal_df["Year"].reset_index(drop="true"), tot_stu_averages)
plt.title("Average California Math SAT Scores")
plt.xlabel("Year")
plt.ylabel("Average Math SAT Score")
plt.show()

# Mississippi
tot_stu_df = miss_df[["Score Ranges.Between 200 to 300.Math.Total", "Score Ranges.Betwe
tot_stu_score_df = miss_df[["Score Ranges.Between 200 to 300.Math.Total", "Score Ranges
for i, name in enumerate(tot_stu_score_df.columns):
    tot_stu_score_df[name] = tot_stu_score_df[name] * (250 + (i * 100))
tot_stu_averages = tot_stu_score_df.reset_index(drop="true").sum(axis=1) / tot_stu_df.r

plt.plot(miss_df["Year"].reset_index(drop="true"), tot_stu_averages)
plt.title("Average Mississippi Math SAT Scores")
plt.xlabel("Year")
plt.ylabel("Average Math SAT Score")
plt.show()

# Illinois
tot_stu_df = ill_df[["Score Ranges.Between 200 to 300.Math.Total", "Score Ranges.Betwee
tot_stu_score_df = ill_df[["Score Ranges.Between 200 to 300.Math.Total", "Score Ranges.
for i, name in enumerate(tot_stu_score_df.columns):
    tot_stu_score_df[name] = tot_stu_score_df[name] * (250 + (i * 100))
tot_stu_averages = tot_stu_score_df.reset_index(drop="true").sum(axis=1) / tot_stu_df.r

plt.plot(ill_df["Year"].reset_index(drop="true"), tot_stu_averages)
plt.title("Average Illinois Math SAT Scores")
plt.xlabel("Year")
plt.ylabel("Average Math SAT Score")
plt.show()
```


Average California Math SAT Scores
Average Math SAT Score
660
640
620
600
580
560
540
520
2006
2008
2010
2012
2014
Year


Average Mississippi Math SAT Scores
Average Math SAT Score
560
540
520
500
480
460
2006
2008
2010
2012
2014
Year

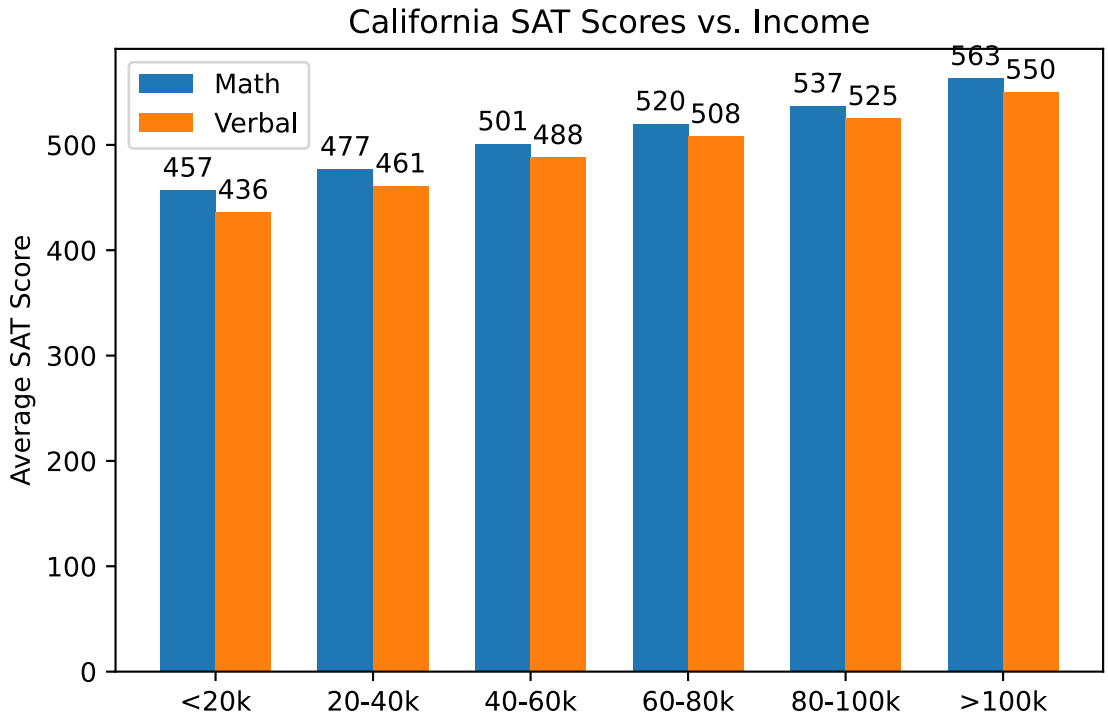


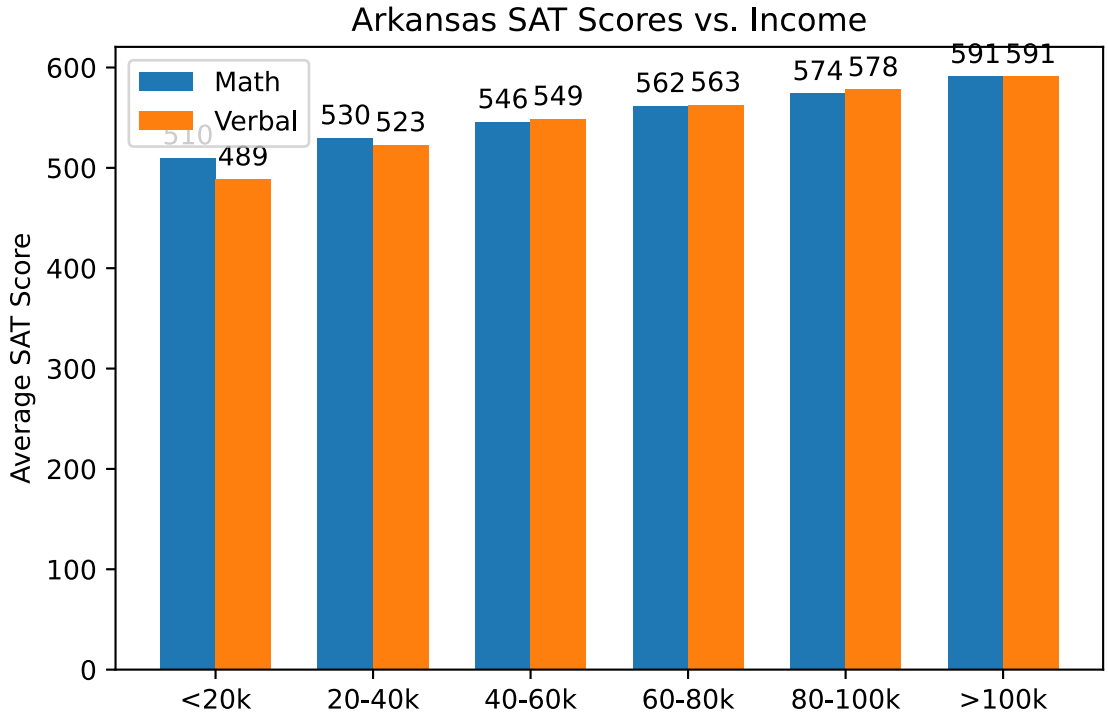
In [3]:

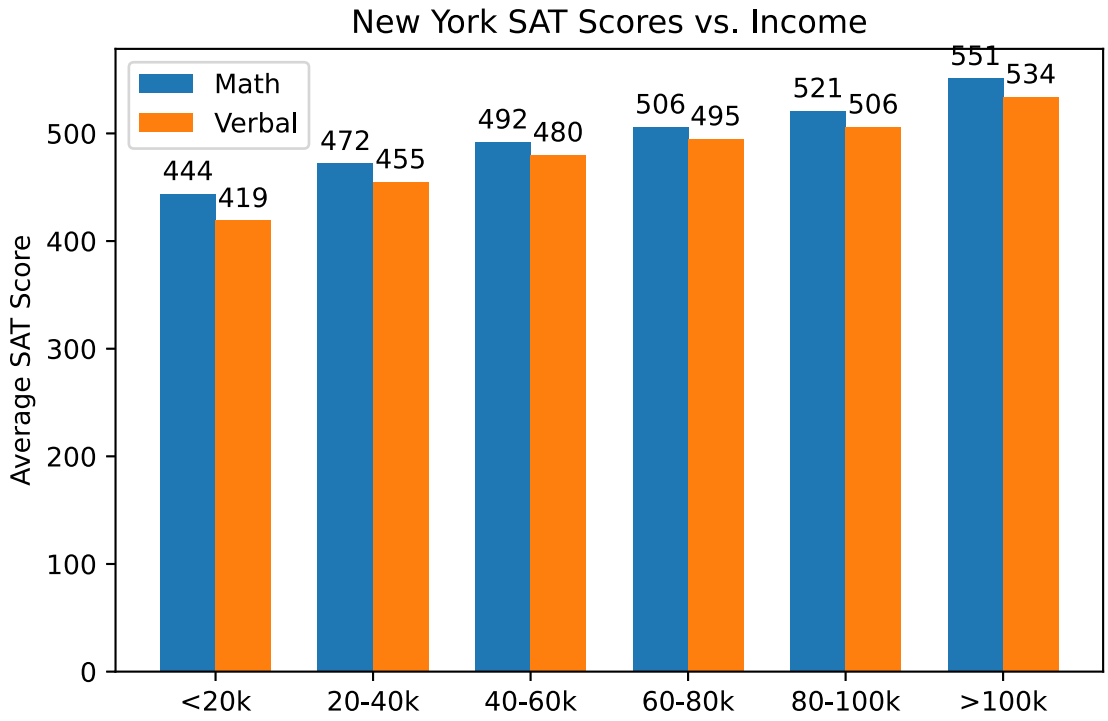
```
# Analysis 2: Income vs SAT score for 4 different states in the SAME year (CA, AR, NY,
# Reason: It seems likely that there would be a strong correlation between family incom

# Pull out state data for 2010
cal_df = corgis_df[corgis_df["State.Name"] == "California"]
ark_df = corgis_df[corgis_df["State.Name"] == "Arkansas"]
new_df = corgis_df[corgis_df["State.Name"] == "New York"]
nor_df = corgis_df[corgis_df["State.Name"] == "North Dakota"]

cal_df = cal_df[cal_df["Year"] == 2010]
ark_df = ark_df[ark_df["Year"] == 2010]
new_df = new_df[new_df["Year"] == 2010]
nor_df = nor_df[nor_df["Year"] == 2010]

cal_df = cal_df.reset_index(drop="true")
ark_df = ark_df.reset_index(drop="true")
new_df = new_df.reset_index(drop="true")
nor_df = nor_df.reset_index(drop="true")

# Plot family income vs average math and verbal SAT score
# CALIFORNIA
math_df = cal_df[["Family Income.Less than 20k.Math", "Family Income.Between 20-40k.Mat
verbal_df = cal_df[["Family Income.Less than 20k.Verbal", "Family Income.Between 20-40k
income_list = ["<20k", "20-40k", "40-60k", "60-80k", "80-100k", ">100k"]

x = np.arange(len(income_list))  # the label locations
width = 0.35  # the width of the bars
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, math_df.loc[0,:], width, label='Math')
rects2 = ax.bar(x + width/2, verbal_df.loc[0,:], width, label='Verbal')
# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Average SAT Score')
ax.set_title('California SAT Scores vs. Income')
ax.set_xticks(x)
ax.set_xticklabels(income_list)
ax.legend()
```

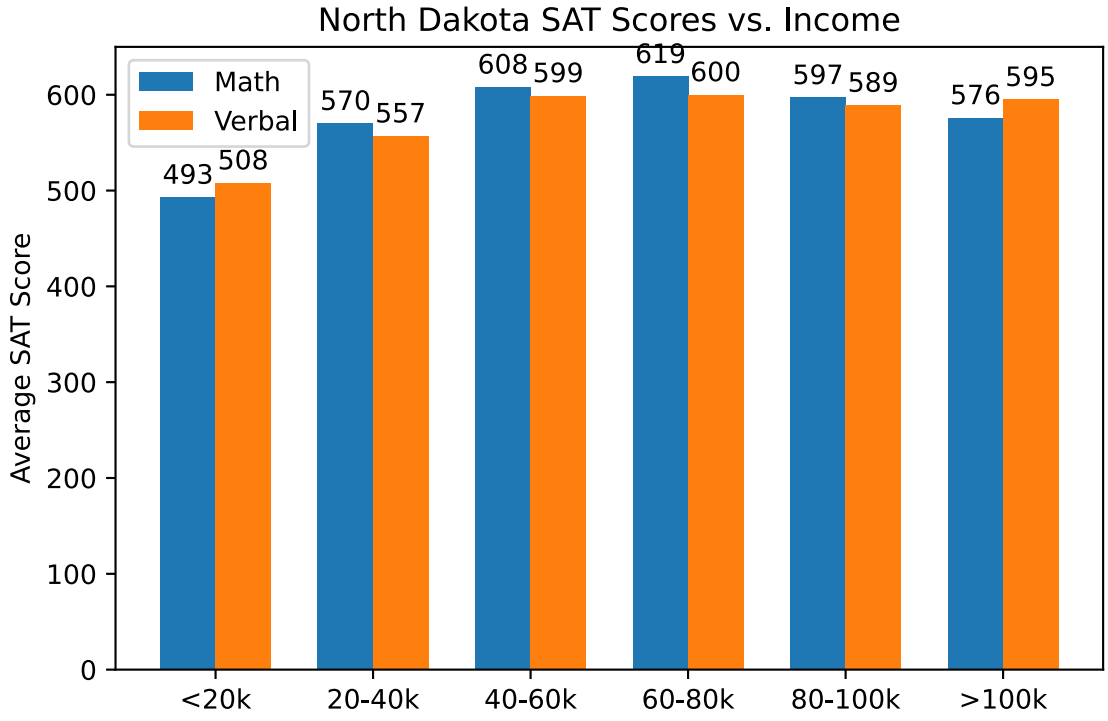
```
def autolabel(rects):
    """Attach a text label above each bar in *rects*, displaying its height."""
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')
autolabel(rects1)
autolabel(rects2)
fig.tight_layout()
plt.show()

# ARKANSAS
math_df = ark_df[["Family Income.Less than 20k.Math", "Family Income.Between 20-40k.Mat
verbal_df = ark_df[["Family Income.Less than 20k.Verbal", "Family Income.Between 20-40k
income_list = ["<20k", "20-40k", "40-60k", "60-80k", "80-100k", ">100k"]

x = np.arange(len(income_list))  # the label locations
width = 0.35  # the width of the bars
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, math_df.loc[0,:], width, label='Math')
rects2 = ax.bar(x + width/2, verbal_df.loc[0,:], width, label='Verbal')
# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Average SAT Score')
ax.set_title('Arkansas SAT Scores vs. Income')
ax.set_xticks(x)
ax.set_xticklabels(income_list)
ax.legend()
def autolabel(rects):
    """Attach a text label above each bar in *rects*, displaying its height."""
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')
autolabel(rects1)
autolabel(rects2)
fig.tight_layout()
plt.show()

# NEW YORK
math_df = new_df[["Family Income.Less than 20k.Math", "Family Income.Between 20-40k.Mat
verbal_df = new_df[["Family Income.Less than 20k.Verbal", "Family Income.Between 20-40k
income_list = ["<20k", "20-40k", "40-60k", "60-80k", "80-100k", ">100k"]

x = np.arange(len(income_list))  # the label locations
width = 0.35  # the width of the bars
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, math_df.loc[0,:], width, label='Math')
rects2 = ax.bar(x + width/2, verbal_df.loc[0,:], width, label='Verbal')
# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Average SAT Score')
ax.set_title('New York SAT Scores vs. Income')
ax.set_xticks(x)
ax.set_xticklabels(income_list)
ax.legend()
def autolabel(rects):
```

```
    """Attach a text label above each bar in *rects*, displaying its height."""
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')
autolabel(rects1)
autolabel(rects2)
fig.tight_layout()
plt.show()

# North Dakota
math_df = nor_df[["Family Income.Less than 20k.Math", "Family Income.Between 20-40k.Mat
verbal_df = nor_df[["Family Income.Less than 20k.Verbal", "Family Income.Between 20-40k
income_list = ["<20k", "20-40k", "40-60k", "60-80k", "80-100k", ">100k"]

x = np.arange(len(income_list))  # the label locations
width = 0.35  # the width of the bars
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, math_df.loc[0,:], width, label='Math')
rects2 = ax.bar(x + width/2, verbal_df.loc[0,:], width, label='Verbal')
# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Average SAT Score')
ax.set_title('North Dakota SAT Scores vs. Income')
ax.set_xticks(x)
ax.set_xticklabels(income_list)
ax.legend()
def autolabel(rects):
    """Attach a text label above each bar in *rects*, displaying its height."""
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')
autolabel(rects1)
autolabel(rects2)
fig.tight_layout()
plt.show()
```


California SAT Scores vs. Income
Math
Verbal
Average SAT Score
0
100
200
300
400
500
457
436
477
461
501
488
520
508
537
525
563
550
<20k
20-40k
40-60k
60-80k
80-100k
>100k


Arkansas SAT Scores vs. Income
Math
Verbal
Average SAT Score
0
100
200
300
400
500
600
510
489
530
523
546
549
562
563
574
578
591
591
<20k
20-40k
40-60k
60-80k
80-100k
>100k

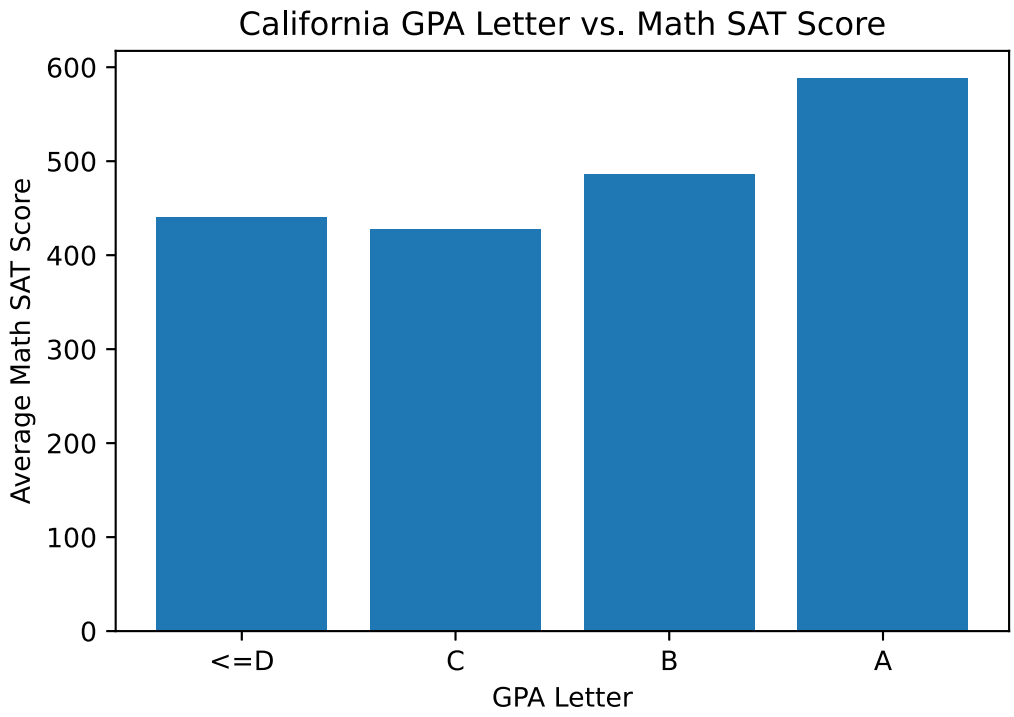


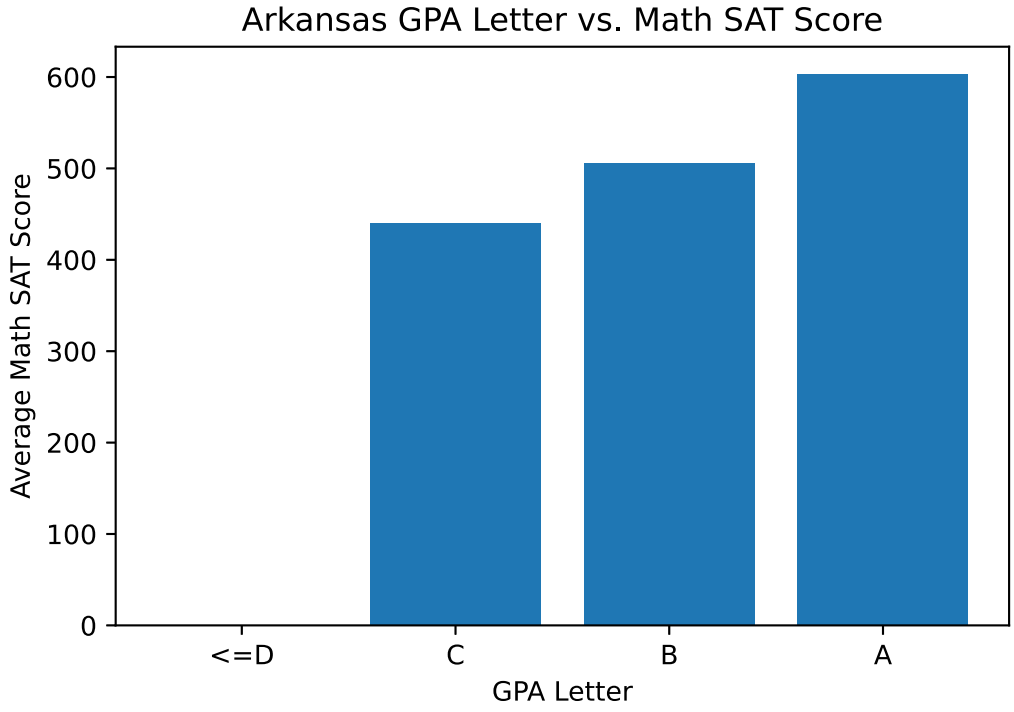


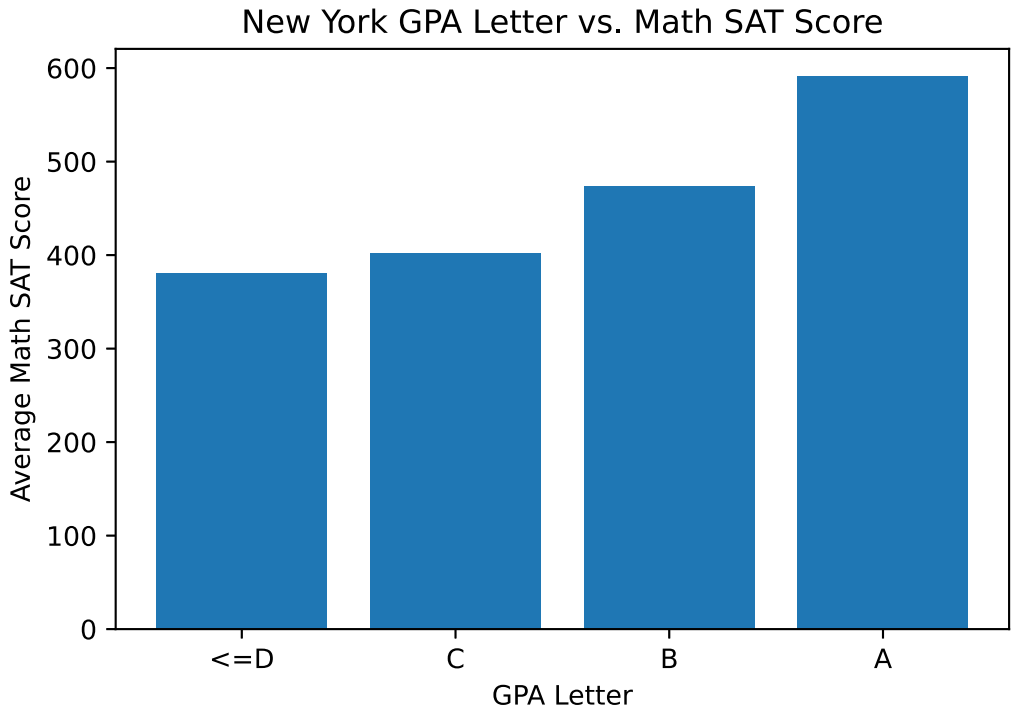
In [4]:

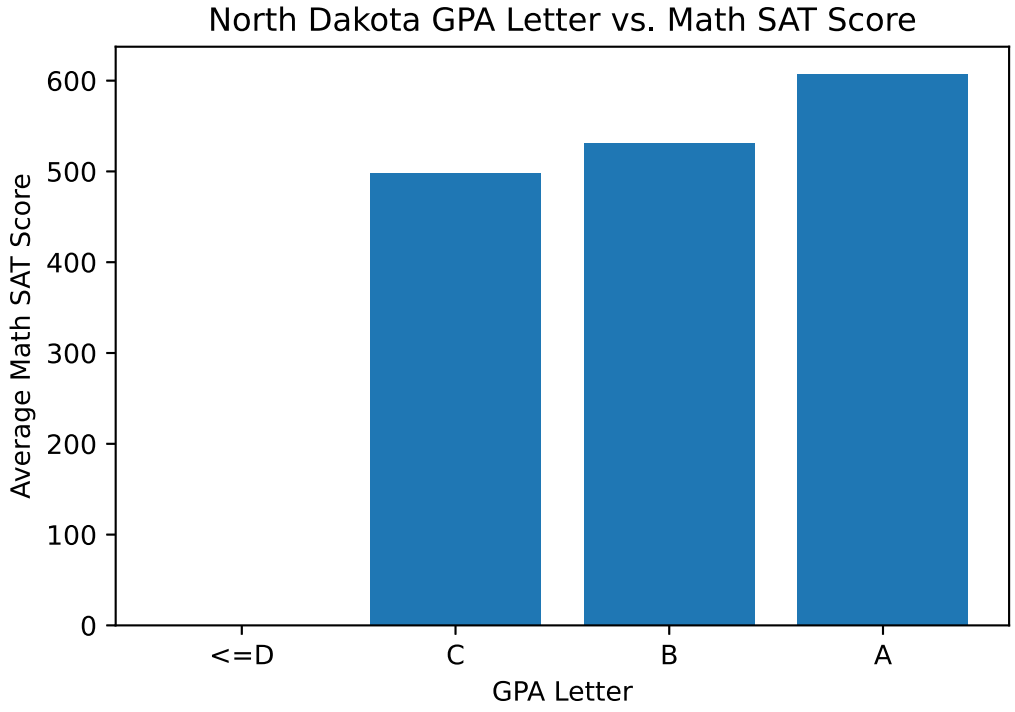
```
# Analysis 3: Sample of GPAs vs SAT Scores
# Reason: It is a logical conclusion that a higher GPA would lead to a higher SAT score

# Pull out state data for 2010
cal_df = corgis_df[corgis_df["State.Name"] == "California"]
ark_df = corgis_df[corgis_df["State.Name"] == "Arkansas"]
new_df = corgis_df[corgis_df["State.Name"] == "New York"]
nor_df = corgis_df[corgis_df["State.Name"] == "North Dakota"]

cal_df = cal_df[cal_df["Year"] == 2010]
ark_df = ark_df[ark_df["Year"] == 2010]
new_df = new_df[new_df["Year"] == 2010]
```

```
nor_df = nor_df[nor_df["Year"] == 2010]

cal_df = cal_df.reset_index(drop="true")
ark_df = ark_df.reset_index(drop="true")
new_df = new_df.reset_index(drop="true")
nor_df = nor_df.reset_index(drop="true")

# Plot GPA vs SAT Score Math
# California
gpa_list = ["<=D", "C", "B", "A"]
avg_gpa_math = [cal_df.at[0, "GPA.D or lower.Math"], cal_df.at[0, "GPA.C.Math"], cal_df
plt.bar(gpa_list, avg_gpa_math)
plt.title("California GPA Letter vs. Math SAT Score")
plt.xlabel("GPA Letter")
plt.ylabel("Average Math SAT Score")
plt.show()

# Arkansas
gpa_list = ["<=D", "C", "B", "A"]
avg_gpa_math = [ark_df.at[0, "GPA.D or lower.Math"], ark_df.at[0, "GPA.C.Math"], ark_df
plt.bar(gpa_list, avg_gpa_math)
plt.title("Arkansas GPA Letter vs. Math SAT Score")
plt.xlabel("GPA Letter")
plt.ylabel("Average Math SAT Score")
plt.show()

# New York
gpa_list = ["<=D", "C", "B", "A"]
avg_gpa_math = [new_df.at[0, "GPA.D or lower.Math"], new_df.at[0, "GPA.C.Math"], new_df
plt.bar(gpa_list, avg_gpa_math)
plt.title("New York GPA Letter vs. Math SAT Score")
plt.xlabel("GPA Letter")
plt.ylabel("Average Math SAT Score")
plt.show()

# North Dakota
gpa_list = ["<=D", "C", "B", "A"]
avg_gpa_math = [nor_df.at[0, "GPA.D or lower.Math"], nor_df.at[0, "GPA.C.Math"], nor_df
plt.bar(gpa_list, avg_gpa_math)
plt.title("North Dakota GPA Letter vs. Math SAT Score")
plt.xlabel("GPA Letter")
plt.ylabel("Average Math SAT Score")
plt.show()

# NOTE: There is data for a few students in Arkansas and North Dakota for the "<=D" cat
```


California GPA Letter vs. Math SAT Score
Average Math SAT Score
600
500
400
300
200
100
0
<=D
C
B
A
GPA Letter


Arkansas GPA Letter vs. Math SAT Score
Average Math SAT Score
600
500
400
300
200
100
0
<=D
C
B
A
GPA Letter

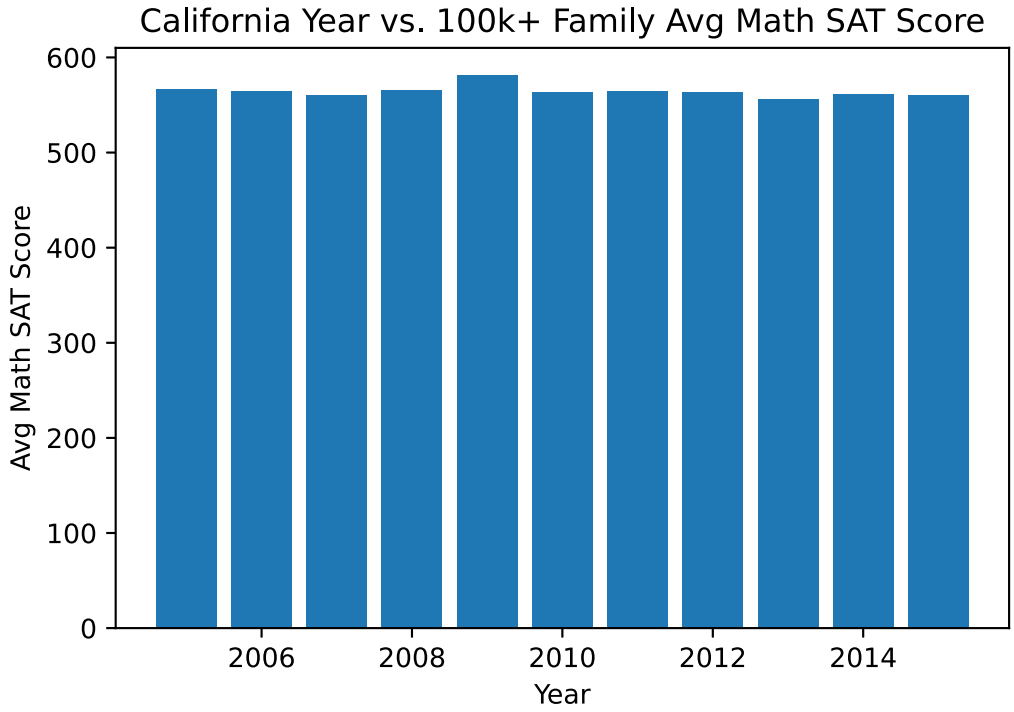


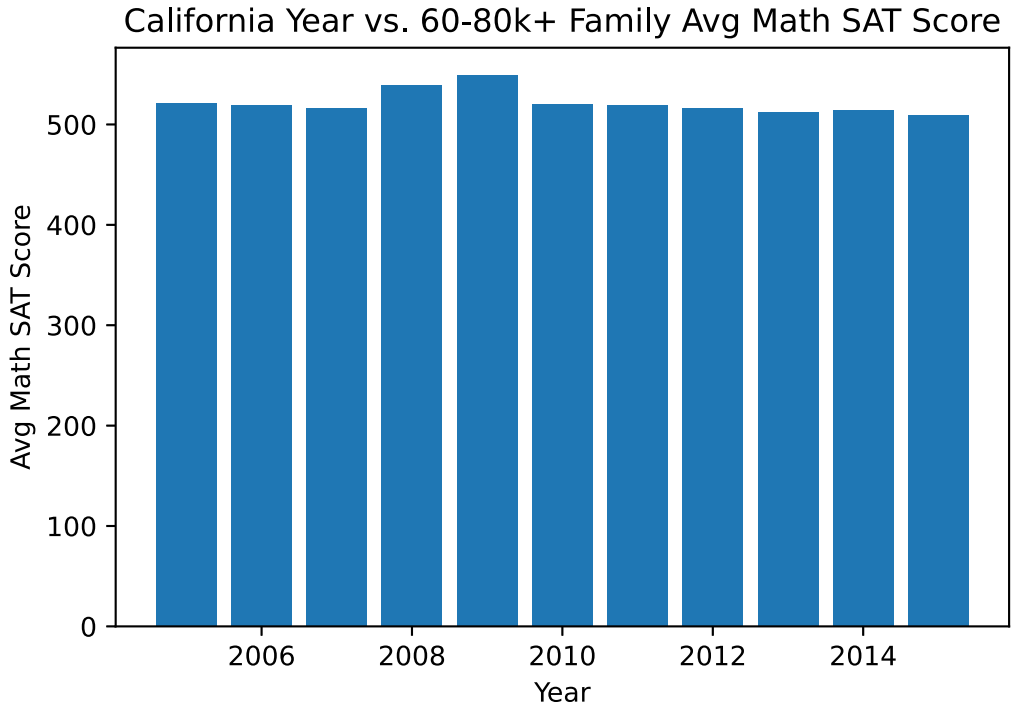


In [5]:

```
# Analysis 4: California High Income vs. SAT Scores over time (2005 - 2015)
# Reason: In order to use more than a single year at a time in our linear regression, w

# Pull in California dataset and restructure indices
cal_df = corgis_df[corgis_df["State.Name"] == "California"]
cal_df = cal_df.reset_index(drop="true")

# Plot bar graph of year vs SAT score for 100k+ earning families
years = np.array(cal_df["Year"])
avg_sat = np.array(cal_df["Family Income.More than 100k.Math"])
plt.bar(years, avg_sat)
plt.title("California Year vs. 100k+ Family Avg Math SAT Score")
plt.xlabel("Year")
```

```
plt.ylabel("Avg Math SAT Score")
plt.show()

# Plot bar graph of year vs SAT score for 60-80k+ earning families
years = np.array(cal_df["Year"])
avg_sat = np.array(cal_df["Family Income.Between 60-80k.Math"])
plt.bar(years, avg_sat)
plt.title("California Year vs. 60-80k+ Family Avg Math SAT Score")
plt.xlabel("Year")
plt.ylabel("Avg Math SAT Score")
plt.show()
```





The School Scores CORGIS Dataset presents a lot of interesting information that reveals social,

educational, and economic challenges that the United States of America faces in regards to college preparedness across the country. There seems to be a strong correlation between economic standing and student success on the SAT from our exploratory analysis without too much discrepancy from year to year. Therefore, depending on our selected variables, our linear regression model will be able to use every data point despite differences in year.

Research Question

Can you predict a US student's Math SAT Score based on their Mathematics GPA and their Music GPA?

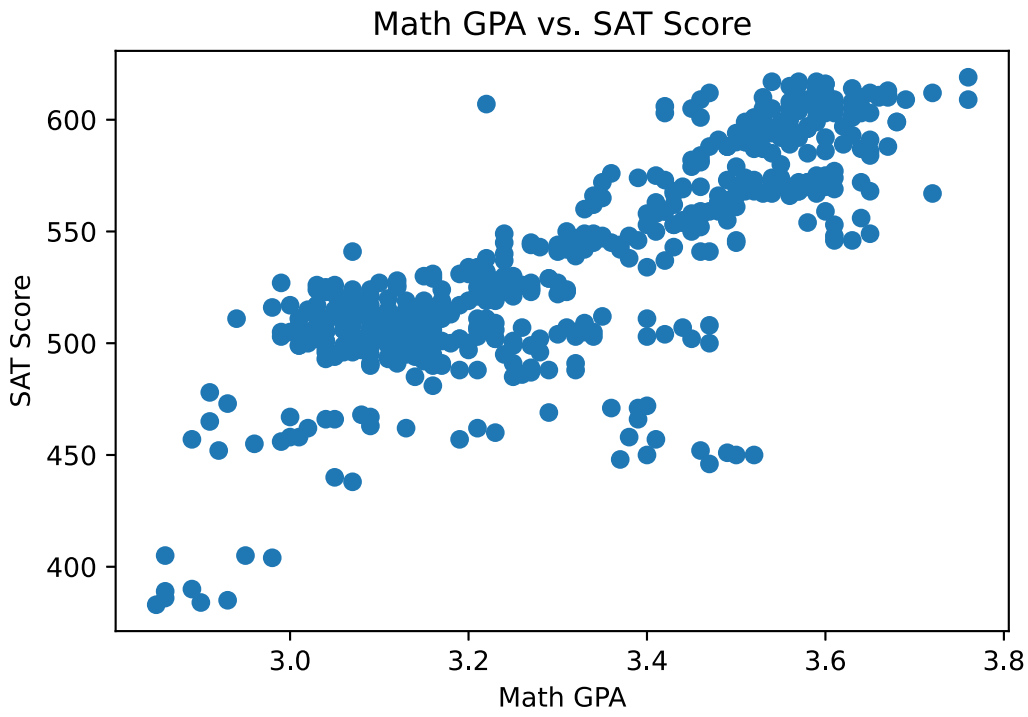
Methods

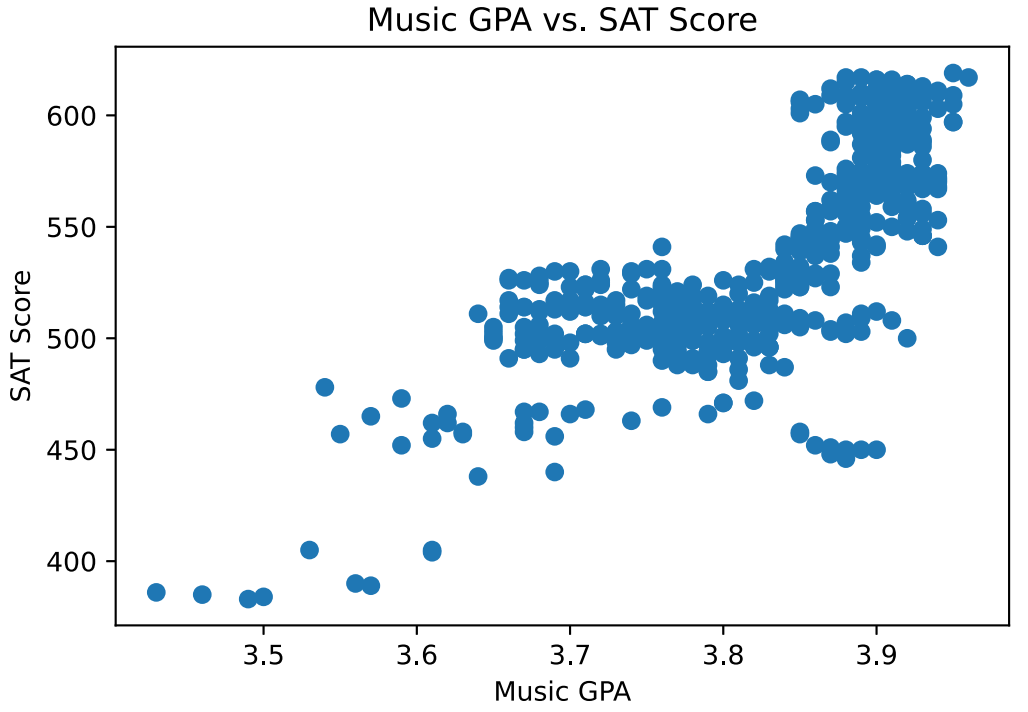
The method we chose analyze our dataset and answer our research question is linear regression. The reason for this choice stems from the fairly linear relationships observed in our exploratory analyses shown above. Now we need to determine whether the use of multiple features can establish a more accurate model that generalizes beyond the training data creates insights into this extensive set of data.

We'll begin by exploring the exact data we'll be using to train the model. Then we will randomize the ordering of the data, split the data into training and testing sets (80/20), and train the model.

In [6]:

```
# Linear Regression Model 1: math_sat_score = w0 + w1*math_gpa + w2*music_gpa

# Explore the variables we will be using relative to SAT scores
# Math GPA
X = np.array(corgis_df["Academic Subjects.Mathematics.Average GPA"]).reshape(corgis_df.
Y = np.array(corgis_df["Total.Math"]).reshape(corgis_df.shape[0], 1)
plt.scatter(X, Y)
plt.title("Math GPA vs. SAT Score")
plt.xlabel("Math GPA")
plt.ylabel("SAT Score")
plt.show()

# Music GPA
X = np.array(corgis_df["Academic Subjects.Arts/Music.Average GPA"]).reshape(corgis_df.s
Y = np.array(corgis_df["Total.Math"]).reshape(corgis_df.shape[0], 1)
plt.scatter(X, Y)
plt.title("Music GPA vs. SAT Score")
plt.xlabel("Music GPA")
plt.ylabel("SAT Score")
plt.show()
```





In [7]:

```
# Get the data ready and begin training the model
X = np.array(corgis_df["Academic Subjects.Mathematics.Average GPA"]).reshape(corgis_df.
X = np.hstack((X, np.array(corgis_df["Academic Subjects.Arts/Music.Average GPA"]).resha
# Add bias
X = np.hstack((np.ones((X.shape[0], 1)), X))

# Append SAT score (label) before shuffle to correctly correlate data rows and shuffle
X = np.hstack((X, np.array(corgis_df["Total.Math"]).reshape(corgis_df.shape[0], 1)))
np.random.shuffle(X)

X_train = X[0:int(X.shape[0] * .80)]
X_test = X[int(X.shape[0] * .80):X.shape[0]]
```

```
Y_train = X_train[:, 3]
Y_test = X_test[:, 3]

# Create test and train data sets
Y_train1 = X_train[:, X_train.shape[1]-1:X_train.shape[1]]
X_train1 = X_train[:, 0:X_train.shape[1]-1]

Y_test1 = X_test[:, X_test.shape[1]-1:X_test.shape[1]]
X_test1 = X_test[:, 0:X_test.shape[1]-1]

# Find weights
W1 = np.linalg.lstsq(X_train1, Y_train1, rcond=None)[0]

# Print out weights and equation:
print("Model 1: math_sat_score = {} + {}*math_gpa + {}*music_gpa".format(W1[0], W1[1],
```

```
Model 1: math_sat_score = [-395.21776422] + [128.28079375]*math_gpa + [132.69110418]*mus
ic_gpa
```

In [8]:

```
# Linear Regression Model 2: math_sat_score = w0 + w1*math_gpa

# Get the data ready and begin training the model
X = np.array(corgis_df["Academic Subjects.Mathematics.Average GPA"]).reshape(corgis_df.
# Add bias
X = np.hstack((np.ones((X.shape[0], 1)), X))

# Append SAT score (label) before shuffle to correctly correlate data rows and shuffle
X = np.hstack((X, np.array(corgis_df["Total.Math"]).reshape(corgis_df.shape[0], 1)))
np.random.shuffle(X)

# Create test and train data sets
Y_train2 = X_train[:, X_train.shape[1]-1:X_train.shape[1]]
X_train2 = X_train[:, 0:2]

Y_test2 = X_test[:, X_test.shape[1]-1:X_test.shape[1]]
X_test2 = X_test[:, 0:2]

# Find weights
W2 = np.linalg.lstsq(X_train2, Y_train2, rcond=None)[0]

# Print out weights and equation:
print("Model 2: math_sat_score = {} + {}*math_gpa".format(W2[0], W2[1]))
```

```
Model 2: math_sat_score = [-51.80710101] + [177.77622533]*math_gpa
```

In [9]:

```
# Linear Regression Model 3: math_sat_score = w0 + w1*music_gpa

# Get the data ready and begin training the model
X = np.array(corgis_df["Academic Subjects.Arts/Music.Average GPA"]).reshape(corgis_df.s
# Add bias
X = np.hstack((np.ones((X.shape[0], 1)), X))

# Append SAT score (label) before shuffle to correctly correlate data rows and shuffle
X = np.hstack((X, np.array(corgis_df["Total.Math"]).reshape(corgis_df.shape[0], 1)))
np.random.shuffle(X)

# Create test and train data sets
Y_train3 = X_train[:, X_train.shape[1]-1:X_train.shape[1]]
X_train3 = X_train[:, [0, 2]]
```

```
Y_test3 = X_test[:, X_test.shape[1]-1:X_test.shape[1]]
X_test3 = X_test[:, [0, 2]]

# Find weights
W3 = np.linalg.lstsq(X_train3, Y_train3, rcond=None)[0]

# Print out weights and equation:
print("Model 3: math_sat_score = {} + {}*music_gpa".format(W3[0], W3[1]))
```

```
Model 3: math_sat_score = [-983.79740905] + [397.75669016]*music_gpa
```

Results

We created 3 different models to be tested and compare their SSE to decide which model will be the best to solve for a student's Math SAT Score based on either their Math GPA, Music GPA, or both.

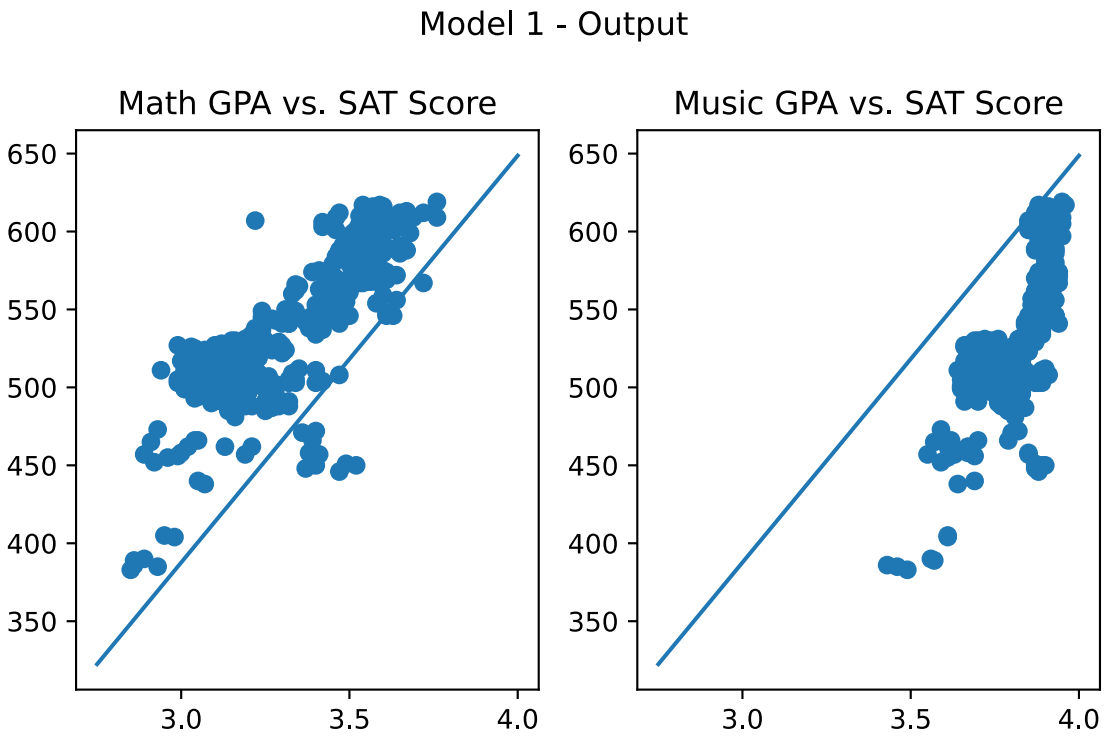
- Model 1: Math_SAT_Score = W0 + W1*Math_GPA + W2*Music_GPA
- Model 2: Math_SAT_Score = W0 + W1*Math_GPA
- Model 3: Math_SAT_Score = W0 + W1*Music_GPA

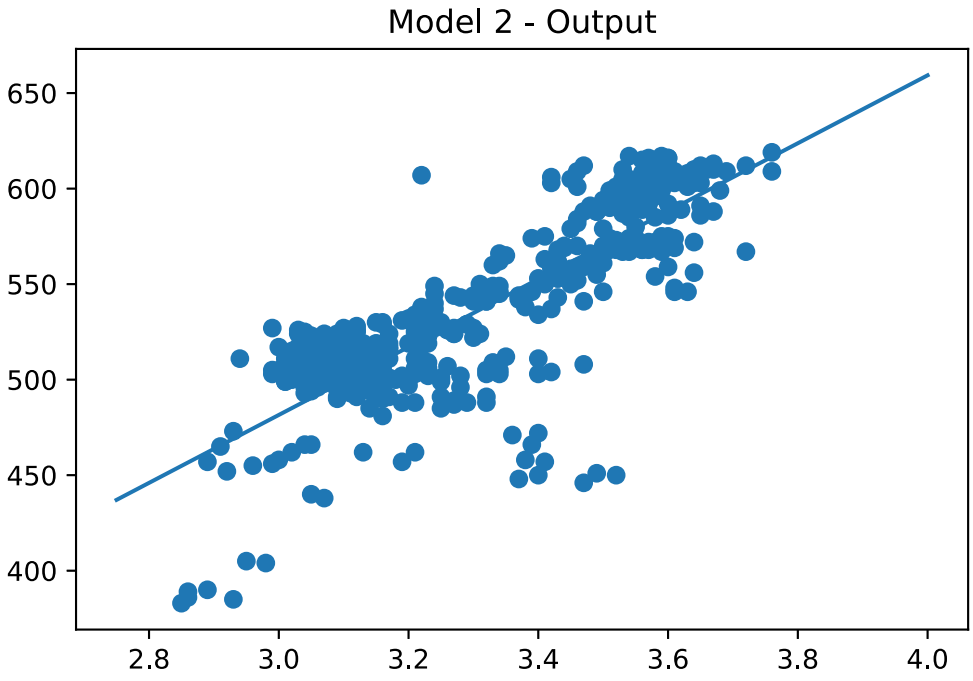
Model Plots and SSE Before Picking Best Model

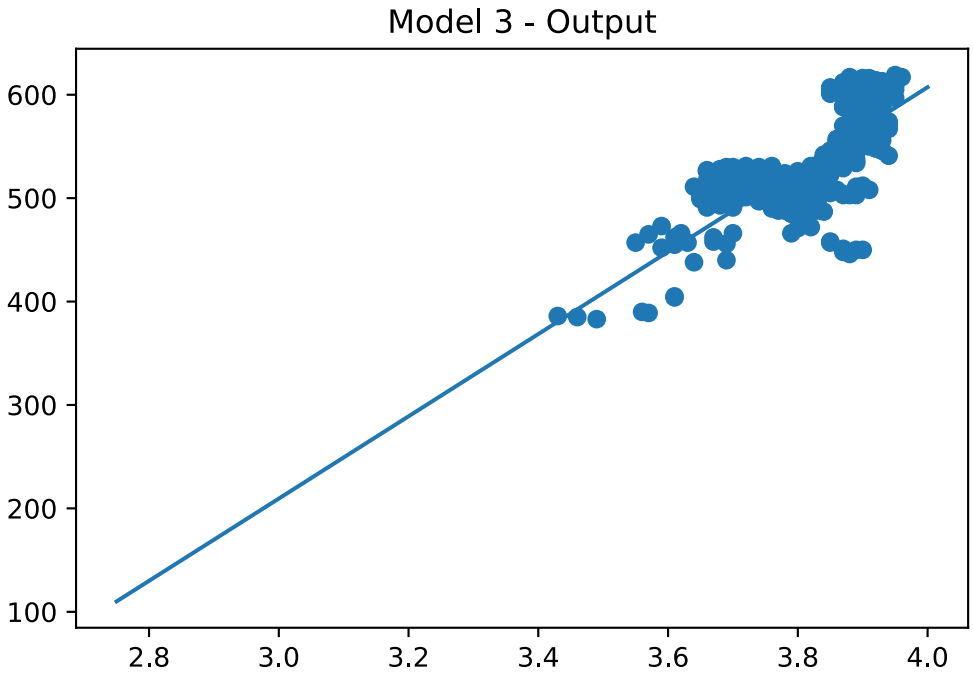
In [10]:

```
# Plot the model in 2-dimensions for both parameters
fig, ax = plt.subplots(1, 2)
fig.tight_layout()

math_vals = np.linspace(2.75, 4)
music_vals = np.linspace(2.75, 4)

# Model 1
SAT_vals = W1[0] + (W1[1]*math_vals) + (W1[2]*music_vals)
#Math
ax[0].scatter(X_train[:, 1], X_train[:, X_train.shape[1]-1])
ax[0].plot(math_vals, SAT_vals)
ax[0].set_title("Math GPA vs. SAT Score")
#Music
ax[1].scatter(X_train[:, 2], X_train[:, X_train.shape[1]-1])
ax[1].plot(music_vals, SAT_vals)
ax[1].set_title("Music GPA vs. SAT Score")

fig.suptitle("Model 1 - Output")
fig.tight_layout()
plt.show()

# Model 2
SAT_vals = W2[0] + (W2[1]*math_vals)
#Math
plt.scatter(X_train[:, 1], X_train[:, X_train.shape[1]-1])
plt.plot(math_vals, SAT_vals)
plt.title("Model 2 - Output")
plt.show()

# Model 3
SAT_vals = W3[0] + (W3[1]*music_vals)
#Math
```

```
plt.scatter(X_train[:, 2], X_train[:, X_train.shape[1]-1])
plt.plot(music_vals, SAT_vals)
plt.title("Model 3 - Output")
plt.show()


# Calculate Training SSE for Model 1
pred_y_1 = W1[0] + (W1[1]*X_train[:, 1]) + (W1[2]*X_train[:, 2])
SSE1 = 0
for i in range(len(X_train)):
    SSE1 += (pred_y_1[i] - X_train[i, X_train.shape[1]-1])**2
print("Model 1 Training SSE: {}".format(SSE1))

# Calculate Training SSE for Model 2
pred_y_2 = W2[0] + (W2[1]*X_train[:, 1])
SSE2 = 0
for i in range(len(X_train)):
    SSE2 += (pred_y_2[i] - X_train[i, X_train.shape[1]-1])**2
print("Model 2 Training SSE: {}".format(SSE2))

# Calculate Training SSE for Model 1
pred_y_3 = W3[0] + (W3[1]*X_train[:, 2])
SSE3 = 0
for i in range(len(X_train)):
    SSE3 += (pred_y_3[i] - X_train[i, X_train.shape[1]-1])**2
print("Model 3 Training SSE: {}".format(SSE3))
```







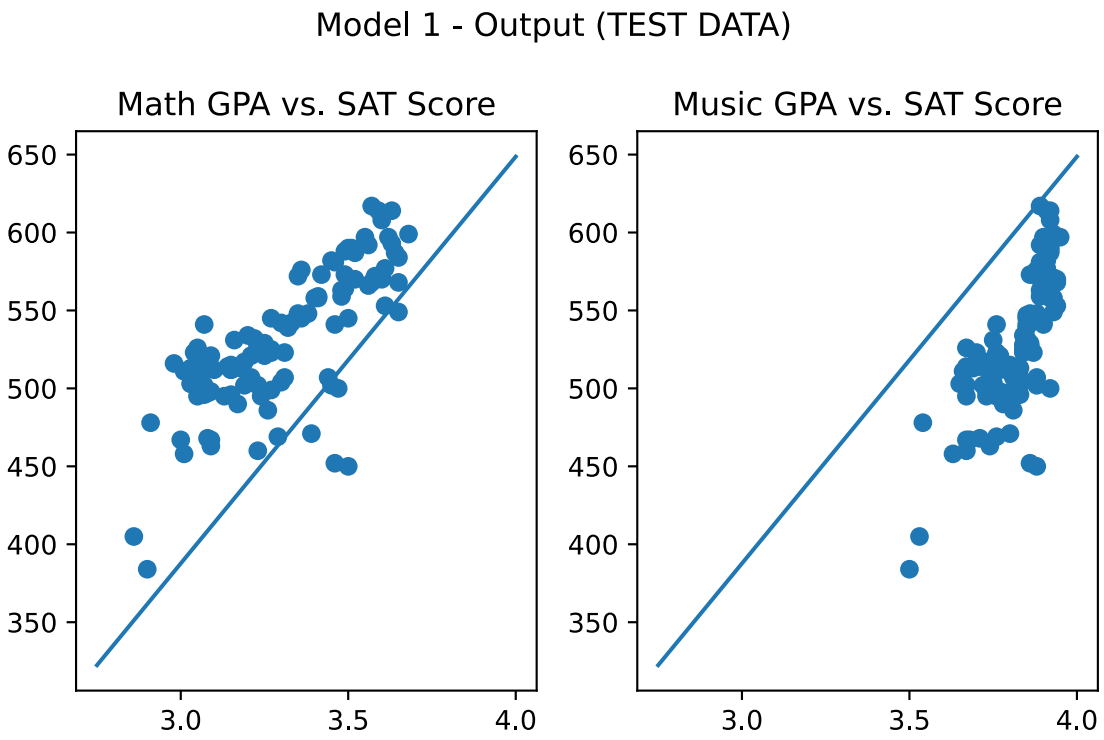

```
Model 1 Training SSE: 306158.9009913763
Model 2 Training SSE: 321957.9831833286
Model 3 Training SSE: 387956.3800788472
```

Model Selection

The model we selected to best represent the data and answer our research question was **Model 1**. Model 1 had a lower training error than the other two models and also seemed to generalize better when looking at 2D plots of the regression line. Now we will plot the line vs the testing data and check the RMSE for those points to see if this model does in fact answer our research question of whether you can predict a student's math SAT score based on their math and music GPAs.

In [11]:
```
# Use test data in trained Model 1
```

```
fig, ax = plt.subplots(1, 2)
math_vals = np.linspace(2.75, 4)
music_vals = np.linspace(2.75, 4)
SAT_vals = W1[0] + (W1[1]*X_test[:, 1]) + (W1[2]*X_test[:, 2])
line_vals = W1[0] + (W1[1]*math_vals) + (W1[2]*music_vals)
#Math
ax[0].scatter(X_test[:, 1], X_test[:, X_test.shape[1]-1])
ax[0].plot(math_vals, line_vals)
ax[0].set_title("Math GPA vs. SAT Score")
#Music
ax[1].scatter(X_test[:, 2], X_test[:, X_test.shape[1]-1])
ax[1].plot(music_vals, line_vals)
ax[1].set_title("Music GPA vs. SAT Score")

fig.suptitle("Model 1 - Output (TEST DATA)")
fig.tight_layout()
plt.show()
```



In [12]:

```
# Calculate RMSE of Model 1 on Testing data
RMSE1 = 0
for i in range(len(Y_test)):
    RMSE1 += ((SAT_vals[i] - Y_test[i])**2) / len(Y_test)
RMSE1 = math.sqrt(RMSE1)
print("Model 1 test RMSE: {}".format(RMSE1))
```

```
Model 1 test RMSE: 28.002443405627005
```

Discussion

The research question we set out to answer was answered in the affirmative! Our trained model used the equation Math_SAT_Score = W0 + W1*Math_GPA + W2*Music_GPA and had a RMSE of roughly 27.7. Since the RMSE is roughly translated to how much each point calculated is off the

correct answer on average, 27.7 is an acceptable amount of error when the SAT Math scores range from 0-800; which means 27.7 is a ~3% of the total values.

Our model does not completely and unfallably give a definitive answer for students but instead paints a high-level picture of how students will perform on average given their math and music GPAs. The main reason for this how the dataset provides this student data. It doesn't give samples on a per student basis but instead as one sample per state per year over the course of ~11 years (2005-2015; inclusive). Each state sample is a collection of averages and composite values that don't allow for the granual analysis that linear regression thrives on. Thankfully, we were still able to use linear regression on this data set due to the strong correlation between math and music GPAs and Math SAT scores across states and years (as shown in our exploratory analysis).

Future work with this dataset would greatly benefit from an unraveling of the averages and a move towards individual, anonomized student scores and data to be analyzed. As it stands, the dataset doesn't allow for too much in-depth analysis due to its averaged nature. However, our outcome can be used by education administrative bodies to understand that math alone is not the only factor in achieving a high math SAT score. Music GPAs contribute to a student's performance with a high correlation. So when schools cut music programs they are not only removing an avenue of creative expression but a laboratory of intellectual cultivation that extends far beyond the task at hand.