Lab 5 Report
Calvin Schaul
IDSN542 Machine Intelligence

Dataset Selection

For this lab, my goal was to find a dataset that connected to my interests and knowledge of computer graphics. Following the given instructions and searching through Kaggle's datasets with the "regression" keyword, I selected a dataset from University of California Irvine's Machine Learning repository that examined over 200,000 possible parameters for matrix multiplication.

The Single Precision General Matrix Multiplication (SGEMM) dataset collected results measuring a GPU's time to do matrix multiplication based on its input parameters - such as numeric data detailing the size of smaller sub-matricies and memory allocations as well as categorical data on other calculation settings.

Data Preparation

Based on our Chapter Two lectures, the following steps were used to prepare this data

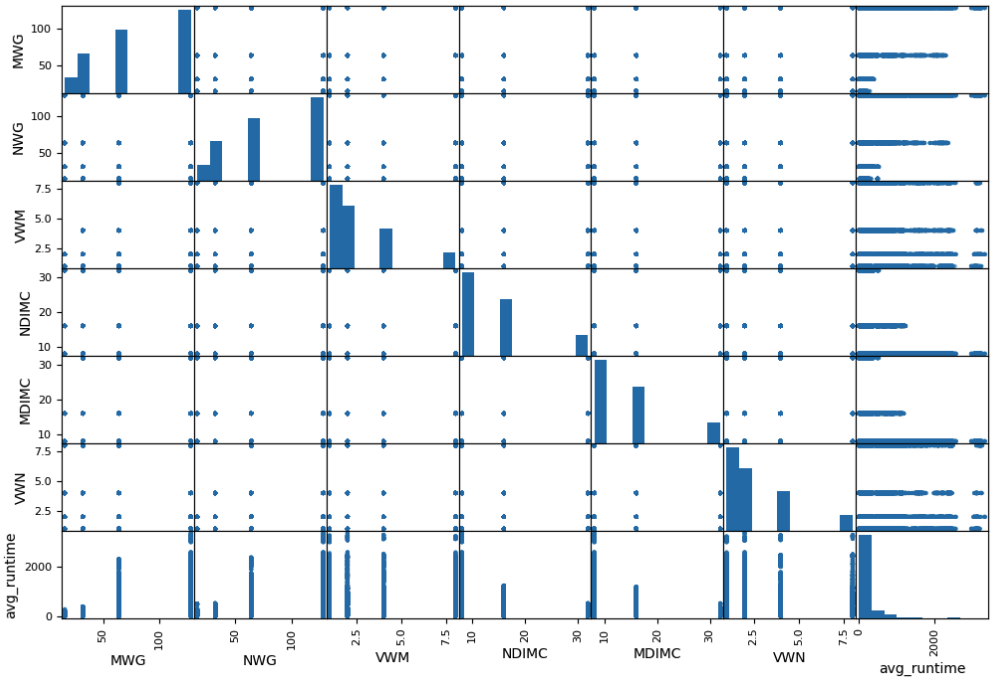
1. Verification that there was no missing data
    a. Externally in a spreadsheet, all data was confirmed to be present
    b. Inside of the code, a test on the number of "NaN"s in the data was run. No attributes were missing information
2. Further analysis based on expert information
    a. On the dataset's Kaggle page and on the UCI repository, two notes were given to analyze this data based on average runtime as well as a logarithm of the runtime.
        i. This relates directly to our points in class about speaking with clients and experts on a subject to better understand the data.
    b. These basic calculations were run after loading in the data from a CSV and added as "avg_runtime" and "log_avg_runtime" columns
    c. An index column was also added based on examples
3. Randomized and Stratified Split
    a. To better understand these two approaches that were discussed in class, the data was split into testing and training data both randomly and using Scikit Learn's Stratified split.
4. Categorical Encoding

a. This dataset indicated that four of the attributes were categorical. These were parameter settings for the GPU kernel and were listed as a “0” or “1” indicating whether the setting was on or off.
b. Once the data was split and we were working with training data, One Hot encoding was used to process the categorical attributes.
    i. Reasoning is below in the “Why One Hot Encoding” section below

5. Correlation and Plotting
    a. With all of the attributes living in a nice training dataset, correlation was calculated to determine which attributes directly correlated with the average runtime.
    b. Scatter plots based on our class examples helped visualize the relationship between highly correlated attributes.
        i. Some further exploration was done with hex bin graphs to try and visualize the density of runtime results and which GPU parameters were frequently performing well.

Correlation Graph (and coefficients)

The correlation graph below shows the relationships across the most correlated numeric attributes in the dataset (|coefficients| > 0.1). The results here are interesting and differ from our example housing data because of the discrete values that the GPU kernel settings. This basic scatter plot makes it challenging to directly visualize **how many** samples performed well (low avg_runtime).

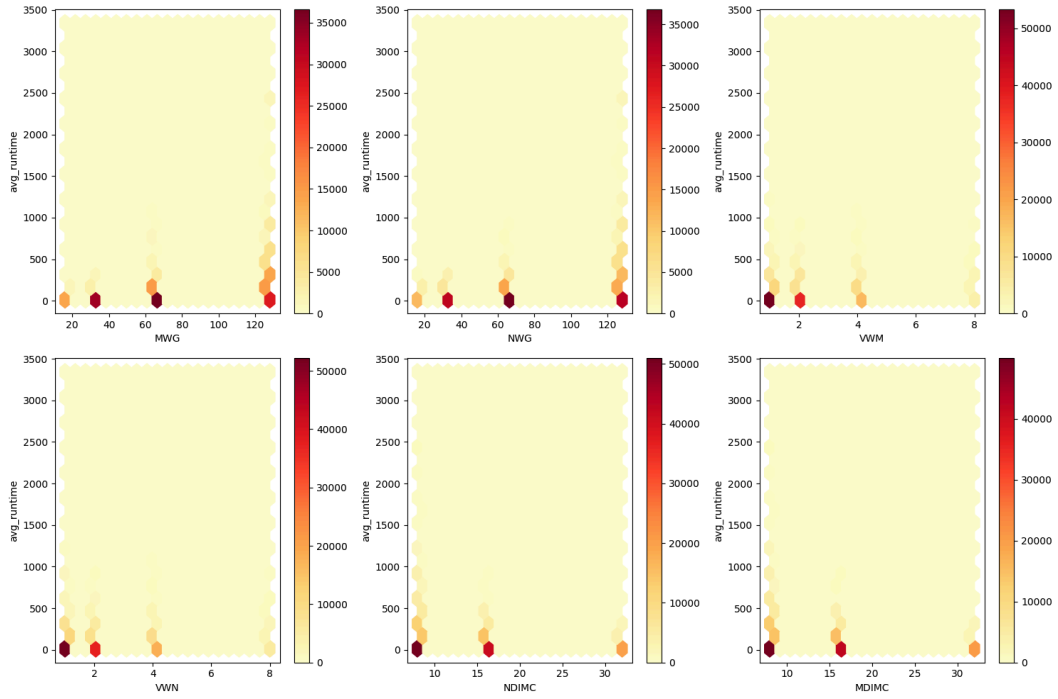


For further reference, here were the correlation coefficients of the attributes:

***Bolded entries were input attributes that had an absolute value correlation coefficient >0.1. These attributes were used for plotting**

avg_runtime	1.000000
Run2 (ms)	0.999991
Run4 (ms)	0.999990
Run3 (ms)	0.999990
Run1 (ms)	0.999970
log_avg_runtime	0.809934
index	0.412433
MWG	**0.351515**
NWG	**0.321099**
VWM	**0.164977**
VWN	**0.145044**
KWI	0.032363
KWG	0.011169
NDIMB	-0.007840
MDIMA	-0.008124
NDIMC	**-0.214328**
MDIMC	**-0.221303**

Finally, a test was done using AI coding tools to help generate a plot to show density of the number of samples and their runtimes, similarly to the way we plotted population in our Chapter 2 Housing example. AI tools were used here because I was struggling to directly translate the population heatmap from class to my dataset.



Analysis of Linear Relationships

Analyzing the correlation from the dataset, and even further analyzing across logarithmic runtimes, shows that there are several GPU settings that had the most impact on runtime performance.

Specifically, the smaller matrix dimensions (MWG, NWG0 used within the larger matrix calculation had the most direct impact. These were even further correlated when measuring the log of the runtime. The least correlated attributes were related to memory shape and workgroup sizing.

The attribute meanings above are summarized from the dataset provider. The correlation analysis is more important than the exact technical meaning of this for this lab's purposes.

avg_runtime 1.000000	log_avg_runtime 1.000000
Run2 (ms) 0.999991	Run3 (ms) 0.809959
Run4 (ms) 0.999990	Run4 (ms) 0.809943
Run3 (ms) 0.999990	avg_runtime 0.809934
Run1 (ms) 0.999970	Run2 (ms) 0.809925
log_avg_runtime 0.809934	Run1 (ms) 0.809862
index 0.412433	index 0.511680
MWG 0.351515	**MWG 0.458786**
NWG 0.321099	**NWG 0.346642**
VWM 0.164977	**VWM 0.210028**
VWN 0.145044	**VWN 0.145032**
KWI 0.032363	KWI -0.010295
KWG 0.011169	KWG -0.020433
NDIMB -0.007840	MDIMA -0.023548
MDIMA -0.008124	NDIMB -0.033832
NDIMC -0.214328	**NDIMC -0.242935**
MDIMC -0.221303	**MDIMC -0.252203**

Why One Hot Encoding

One hot encoding was used for this dataset's categorical attributes because they were simply 0's and 1's. This plays perfectly into One Hot Encoding's binary nature.
Ordinal encoding could have been a better choice if there was more text description.

New Attributes

The new attributes created for this dataset were the average runtime and log of the average runtime. These were created based on the dataset provider's guidance to help make it easier to relate a specific GPU setting to a single runtime.

To further test, I also did a base-two logarithm of all of the numerical attributes since they were all powers of two. Since this lab was mainly focused on correlation rather than scaling of data, this was an experiment of scaling data logarithmically. For the next homework assignment I plan on using StandardScalar.