COMP 4522 Assignment 3 Report

Gareth Carvalho

The details of Extraction from MySQL and MongoDB are detailed in my `main.ipynb` in my project folder. The rest of the steps are also detailed in there, but I will delve into it here as well.

Transforming & Data Quality

Each of the relations provided had their own set of problems that needed to be addressed. Some of the relations also had foreign keys to other relations. This made it so that I had to clean and transform some data before I did others.

For example, `Student_Counceling_Information` and `Employee_Information` relations both have `Department_ID` from the `Department_Information` relation as a foreign key. This means that I had to clean `Department_Information` before the other two. A similar situation arose with `Student_Peformance_Data`, where it has the `Student_ID` from the `Student_Information` relation.

Cleaning Data

For the `Department_Information`, I checked whether `Department_ID` and `Department_Name` were missing, and then whether they are unique. I also checked whether `DOE` is missing, and then whether it is a valid year (>=1900). I also store the IDs to use in later cleaning. Any rows that had an issue in their data, I removed from the dataset. Below are the exceptions I've reported in my jupyter notebook:

`Department_Information` exceptions

```
{'index': 22, 'issues': [{'attribute': 'Department_ID', 'issue': 'Not Unique'}, {'attribute': 'DOE', '
{'index': 26, 'issues': [{'attribute': 'Department_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 32, 'issues': [{'attribute': 'DOE', 'issue': 'Invalid Date: 1849'}], 'action': 'Removed'}
{'index': 36, 'issues': [{'attribute': 'Department_ID and Department_Name', 'issue': 'Duplicate Entry'
{'index': 40, 'issues': [{'attribute': 'Department_ID', 'issue': 'Not Unique'}, {'attribute': 'DOE', '
{'index': 46, 'issues': [{'attribute': 'Department_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
```

(I am sorry I had to crop them because the length was too long for them to be readable, they are fully visible in the jupyter notebook)

For `Student_Counceling_Information`, I do a very similar thing to the previous relation in terms of IDs, I check the `Student_ID` to make sure that it isn't missing and that it is unique. I also chop off the `SID` prefix from the ID, and just keep the numbers. I will use that later in the

data analysis/mining. I also check the `Department_Choices` and `Department_Admission` against the IDs I extracted from cleaning the department info. Below are the exceptions.

`Student_Counceling_Information` exceptions

```
{'index': 69, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 74, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 81, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 117, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 142, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 177, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 180, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 226, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 228, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 240, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 263, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 270, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 290, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 299, 'issues': [{'attribute': 'Department_Choices', 'issue': 'Missing'}, {'attribute': 'Dep
{'index': 330, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 352, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 372, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 394, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 428, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 448, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 491, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 493, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 518, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 539, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 568, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
...
{'index': 3771, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 3860, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 3872, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 3975, 'issues': [{'attribute': 'Student_ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
```

Most of them are just non-unique ids. One visible here has its department IDs missing. Any entry with missing information was removed from the dataset.

For `Student_Performance_Data`, a slightly different approach was needed. In the rubric, there is a part saying to look out for duplications of papers from the same student. I wasn't able to find any of those occurrences, but the way I tackled it was as follows:

I had a dictionary of `Student_ID` -> list of `Paper_ID`s. I checked each entry's `Paper_ID` against the list of papers the student already marks for, and would report any that came up as a duplicate (if there was any).

I also tested to make sure every `Student_ID` belonged to an actual student from the `Student_Counceling_Information` relation. I also made sure that there were no missing

values for `Effort_Hours` and `Marks`. Any entries with problematic data were removed from the set. Below are the reported exceptions:

`Student_Performance_Data` exceptions


```
{'index': 329, 'issues': [{'attribute': 'Marks', 'issue': 'Value out of range of 0-100: Value was -49'}], 'action': 'Removed'}
{'index': 415, 'issues': [{'attribute': 'Marks', 'issue': 'Value out of range of 0-100: Value was 207'}], 'action': 'Removed'}
{'index': 552, 'issues': [{'attribute': 'Marks', 'issue': 'Value out of range of 0-100: Value was -100'}], 'action': 'Removed'}
{'index': 841, 'issues': [{'attribute': 'Marks', 'issue': 'Value out of range of 0-100: Value was 140'}], 'action': 'Removed'}
{'index': 59636, 'issues': [{'attribute': 'Effort_Hours', 'issue': 'Invalid amount of hours: Value was -3'}], 'action': 'Removed'}
{'index': 172219, 'issues': [{'attribute': 'Marks', 'issue': 'Missing'}, {'attribute': 'Effort_Hours', 'issue': 'Missing'}], 'action': 'Removed'}
{'index': 181489, 'issues': [{'attribute': 'Marks', 'issue': 'Value out of range of 0-100: Value was 999'}], 'action': 'Removed'}
{'index': 209594, 'issues': [{'attribute': 'Marks', 'issue': 'Missing'}], 'action': 'Removed'}
{'index': 209608, 'issues': [{'attribute': 'Effort_Hours', 'issue': 'Invalid amount of hours: Value was 257'}], 'action': 'Removed'}
```


Cleaning `Employee_Information` was pretty simple. I made sure no data was missing, and made sure that every `Employee_ID` was unique. I also compared the `Department_ID` against the list I extracted from the department relation. Any entry with problematic data was removed.

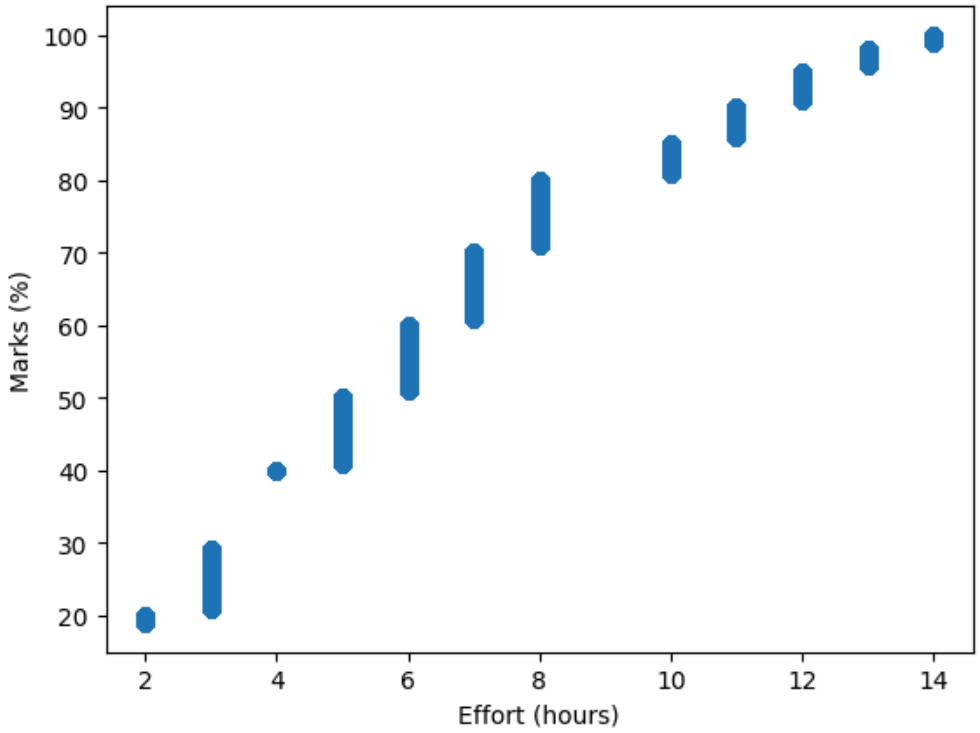
`Employee_Information` exceptions


```
{'index': 313, 'issues': [{'attribute': 'Employee ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
{'index': 890, 'issues': [{'attribute': 'Employee ID', 'issue': 'Not Unique'}], 'action': 'Removed'}
```

Data Mining

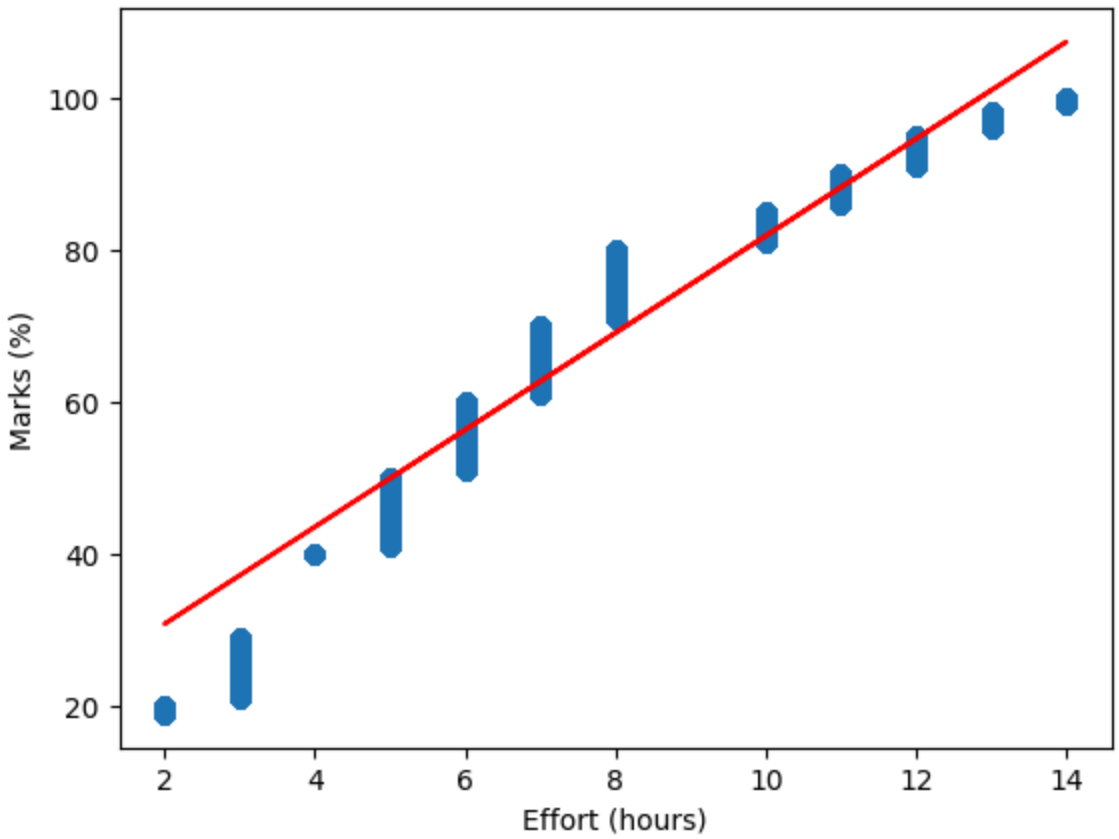
Descriptive Data

The analysis I did drew a correlation between `Effort_Hours` and `Marks` in the `Student_Performance_Table`. I created a scatter plot to demonstrate this.



The correlation is pretty clear, but who is to really say that putting in more effort yields greater marks? Anyway, I set out to do predictive analysis on this data.

I drew a linear regression to predict a student's mark from the effort they put in. The line of best fit looks like this:

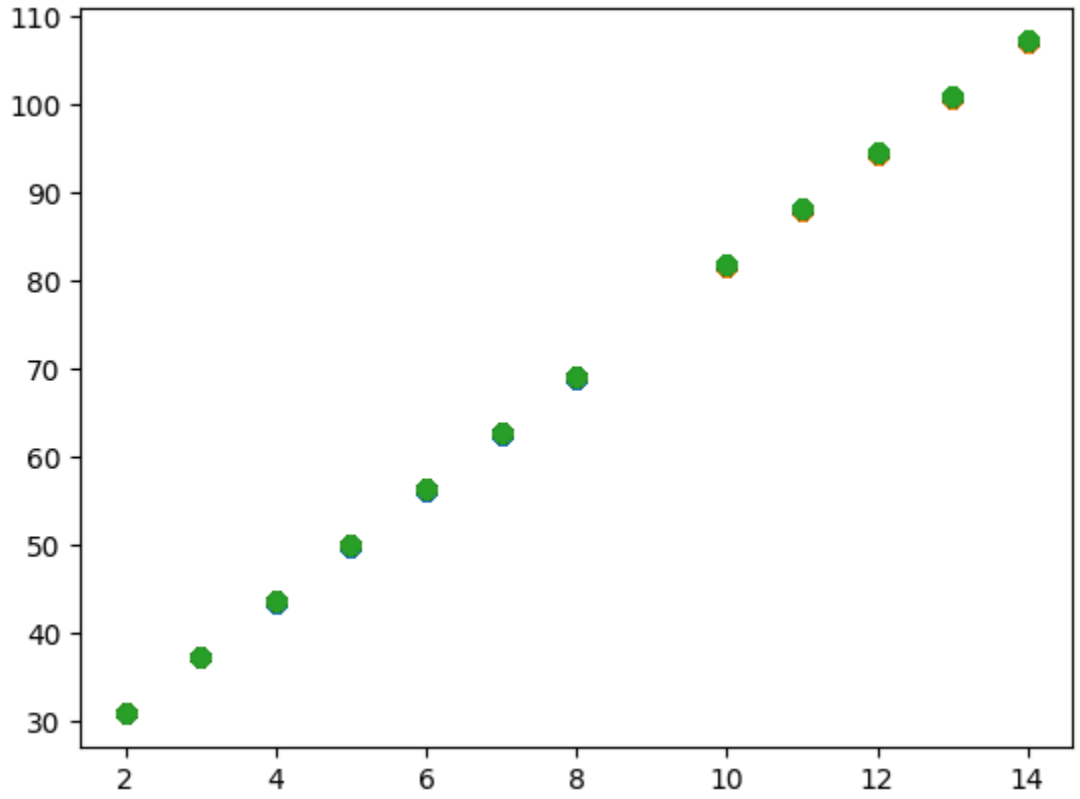


Of course, this regression doesn't actually take into account any specific student, but rather is a line representative of all student's efforts and marks accumulated.

Predictive Data

I created a second model to address this. A multiple linear regression, using Student ID and Effort to predict their marks. My hope was that the model could predict a mark any particular student was going to achieve based on their effort.

After training, I plugged in the three Student IDs from the assignment document, and plotted this:



Obviously, very negligible differences between the marks. In fact, the differences visually are a couple of pixels for the higher amount of hours. The assignment document specifically asked to test their marks with 10 hours of effort, and so the model produced this:

```
Marks for 'SID20131151' for 10 hours: 81.76466775010385
Marks for 'SID20149500' for 10 hours: 81.77677817220525
Marks for 'SID20182516' for 10 hours: 81.79856887993317
```

Again, the difference is extremely negligible.

Conclusion

In conclusion, I actually found the cleaning of the data a lot more intuitive than the model training. Partially because you laid out what we had to clean, but also because there is an intuitive sense of what is important and what isn't when gathering data.

The analysis on the other hand, while fun at times, was a lot more of a hassle in my opinion. I didn't find the libraries easy to use, and found very few tutorials that were actually helpful for

someone at my skill level doing this. Of course, the basics I understand, but sometimes when it gets covered in python syntax and other things I find it easy to get lost. There was some more stuff I wanted to try in regards to analyzing the data, such as looking at average marks per `Paper_ID`, but I wasn't able to figure out how to go about that.