COMP-4522-002 Database II

Assignment 3: Raw Data Processing, ETL, and Data Mining

Aldo Ortiz and Iyan Velji

Dr. Orestes Appel

April 8, 2024

Contents

1. Data Cleaning With Pandas and SQLite

For the data cleaning portion of the assignment, our code loads data from CSV files, validates the data, cleans the data, merges the data, and performs functions like aggregation.

First, our code loads data from four CSV files into Pandas data frames using Python. Each data frame corresponds to a specific domain within the educational institution such as departments, employees, students, and student performance. The data frames are later converted into SQL tables using SQLAlchemy to allow the potential for direct queries.

Next, we validate the data using the validate_clean_data function on each data frame. Our code performs validation by filling in missing values with predefined placeholders to ensure completeness and removing non-numeric characters from the marks to ensure validity. The function is also adaptable to different data frame structures.

The code then merges the loaded data frames based on predefined relationships, creating combined data frames that link the employees to their departments, students to their departments, and students to their performance records. Finally, data aggregation performs a basic aggregation operation, calculating the mean marks for each semester across all students.

Our output prints the heads of various merged and aggregated data frames for quick inspection and troubleshooting.

```
Student Performance DataFrame:
    Student_ID       DOA       DOB  Department_Choices  Department_Admission Semester_Name     Paper_ID  Paper_Name  Marks  Effort_Hours
0   Student_ID       DOA       DOB  Department_Choices  Department_Admission  Semster_Name     Paper_ID  Paper_Name    NaN  Effort_Hours
1  SID20131143  7/1/2013  2/5/1996            IDEPT7783            IDEPT7783         Sem_1  SEMI0012995     Paper 1   44.0             5
2  SID20131143  7/1/2013  2/5/1996            IDEPT7783            IDEPT7783         Sem_1  SEMI0015183     Paper 2   74.0             8
3  SID20131143  7/1/2013  2/5/1996            IDEPT7783            IDEPT7783         Sem_1  SEMI0018371     Paper 3   80.0             8
4  SID20131143  7/1/2013  2/5/1996            IDEPT7783            IDEPT7783         Sem_1  SEMI0015910     Paper 4   44.0             5
```

The head of the student performance data frame is printed for inspection.

2. Data Cleaning With MongoDB

If we had used MongoDB instead of Python to clean, validate, and load our data the process would have been slightly different.

To import the data we would have used the mongoimport tool to get the data from the CSV file in a fast and multi-threaded manner.[1] In addition, mongoimport can be combined with other command-line tools like jq for JSON manipulation or csvkit for directly manipulating CSV files.

```
MongoDB Enterprise > show dbs
admin   0.000GB
local   0.000GB
mydb    0.000GB
MongoDB Enterprise > show collections
MongoDB Enterprise > use mydb
switched to db mydb
MongoDB Enterprise > show collections
AncillaryPhysicians
collector
name
things
MongoDB Enterprise > db.things.find()
MongoDB Enterprise > mongoimport -d mydb -c things --type csv --file sample.csv
--headerline
2016-12-18T10:42:05.272+0530 E QUERY    [main] SyntaxError: missing ; before sta
tement @(shell):1:15
MongoDB Enterprise >
```

An example of monogimport on the command line.

Once the data has been imported, validation could be enforced with schema validation in MongoDB. MongoDB uses a flexible schema model meaning documents do not need to have the same fields or data types by default. Some examples of how schema validation could be used include ensuring passwords are stored as strings,

[1] https://www.mongodb.com/docs/database-tools/mongoimport/

ensuring a product is truly being sold by the store, and ensuring numbers like GPAs are always positive.[2] To ensure consistency, especially in terms of primary and foreign keys. the aggregation framework can be utilized. Additionally, MongoDB ensures that changes made in a transaction are consistent with any constraints.[3] For checking uniqueness, a unique index can be used to ensure that indexed fields do not contain duplicate values.[4] By default, MongoDB creates a unique index on the _id field and new indexes can be added using db.collection.createIndex().

3. Data Mining

Our data mining portion of this assignment was split into two major parts: descriptive analytics and predictive analytics.

Descriptive Analytics

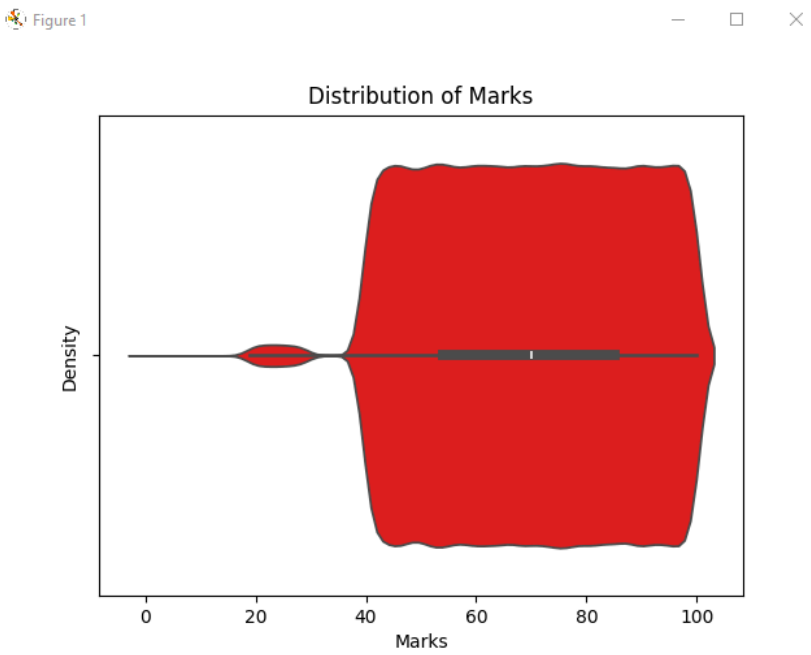
To begin our descriptive analytics portion of the data mining tasks we start by using the describe() function on the student performance and employee data frame. The student performance description returns basic statistical values like the count, mean, standard deviation, minimum, and maximum. The employee description takes a categorical approach displaying the count, unique, top, and frequency of the employee ID, date of birth, DOJ, and department ID.

```
Performance DF Statistical Summary:
               Marks
count  209611.000000
mean       69.588981
std        18.100447
min         0.000000
25%        54.000000
50%        70.000000
75%        85.000000
max       100.000000
Employee DF Statistical Summary:
       Employee_ID        DOB        DOJ Department_ID
count         1001       1001       1001          1001
unique         999        937        921            35
top       IU444477  10/2/1992  9/23/2007     IDEPT5109
freq             2          3          3            42
```

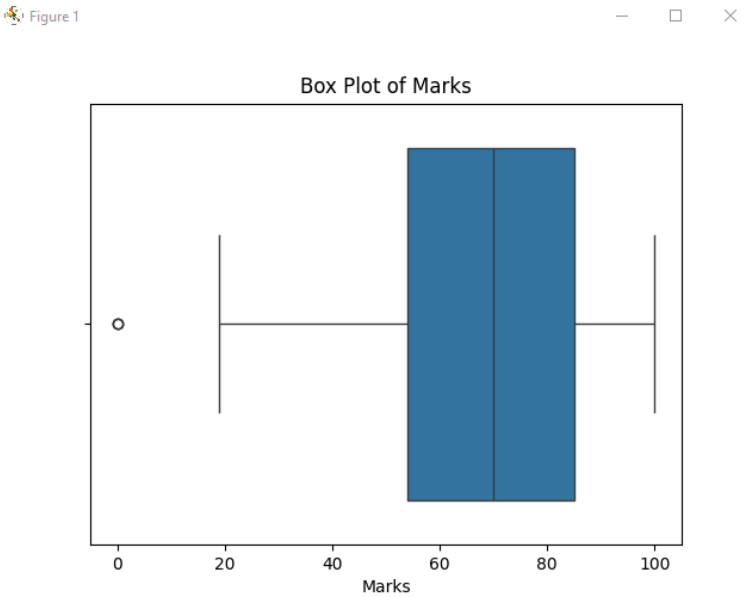
The returned descriptive statistics for the student performance and employee data frame

[2] https://www.mongodb.com/docs/manual/core/schema-validation/
[3] https://www.mongodb.com/docs/manual/data-modeling/data-consistency/
[4] https://www.mongodb.com/docs/manual/core/index-unique/

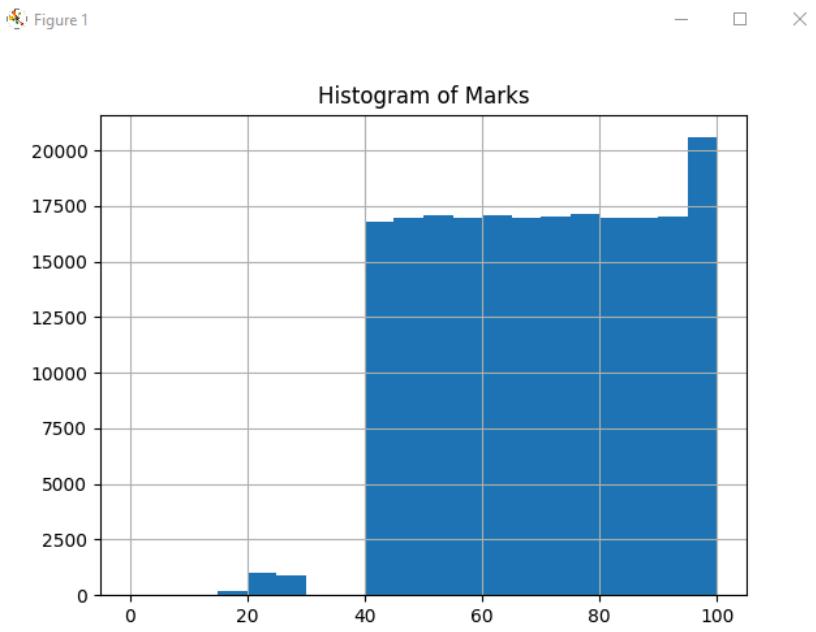
Next, we used Matplotlib and Seaborn to further understand the data from a visual standpoint. We created a histogram, box plot, and violin plot to visualize the distribution, central tendency, spread, and outliers within our dataset.



The violin plot combines aspects of the box plot and histogram to show the distribution of data.



The box plot method allows us to identify the central tendency of the data



The histogram shows the distribution of grades with most students scoring above 40%

Predictive Analytics

To analyze the dataset in a more predictive manner, we used a linear regression model to show the relationship between our dependent variable of marks and our independent variables of effort_hours_next and marks_avg.

First, we started by adding a new column to our performance data frame with a constant value of 10 to represent the number of hours the student is expected to put into their next paper. We then split the data into a training set worth 80% of the data and a testing set worth 20% of the data. The random_state parameter ensures the reproducibility of the results by setting a seed for the random number generator used in the split.

To perform the linear regression analysis, we utilized sci-kit learn on our data frames.

To prove the model works as intended we tested it using the student ID's SID20131151, SID20149500, and SID20182516.

```
  students_avg_marks['Effort_Hours_Next'] = 10
Student ID: SID20131151, Predicted Marks: 72.55
Student ID: SID20149500, Predicted Marks: 66.49
Student ID: SID20182516, Predicted Marks: 71.76
```

Testing our linear regression model

We found that SID20131151 is predicted to have a mark of ~73%, SID20149500 is predicted to have a mark of ~66%, and SID20182516 is predicted to have a mark of ~72%. An eyeball view of the data shows that the predictions appear to be semi-accurate considering the students' past performance.