CSCI-585 Fall 2022 Finals Rubrics

Q1

A. Relational DBs are used to relate data held in different tables. Non-relational DBs do not. List the 4 types of non-relational DBs, briefly explain each in a line or two, making sure to list a USE CASE for each.

Sample Answer:.

- Key-value DBs: Designed for storing, retrieving, and managing associative arrays. k/v DBs are lightweight (simple), schema-less, transaction-less. Example: Memcached, Redis, Amazon's Dynamo USE CASE: to store gameplay data, stock price etc.
- Column family DBs: stored as a group of columns, good for aggregate queries. Examples: BigTable(Google), BigQuery(Google), HBase(Apache), Cassandra(Apache), SimpleDB(Amazon), RedShift(Amazon). USE CASE: for realtime analytics, storing user prefs (which can be different for each row)
- Document DBs: basic unit of storage is a document. Examples: Couchbase, CouchDB, MongoDB, OrientDB. USE CASE: for storing blog posts
- Graph DBs: A graph database uses (contains) graph entities such as nodes (vertices), relations (edges), and properties (k-v pairs) on vertices and edges, to store data. Example: (social networks, recommendation engines), FlockDB (from Twitter), Neo4J, HyperGraphDB, InfiniteGraph, InfoGrid, OrientDB, Giraph (from Apache), GraphLab. USE CASE: for social media, recommendations, etc.

Note - there can be other use cases than the sample ones listed above - so pl. be flexible.
For each type, give 0.5 point if the DB name AND EXPLANATION is given, and 0.5 point if its USE CASE/example is listed.

B. We can talk of a 'time series' (time-stamped data where there will be a 'time' column for sure, to provide a non-null value in each row of data; the goal is to be able to process time-based queries) DB as a 'new' DB type, even though the 5 DB types we studied can be used to store time-stamped data (eg. stock prices, earthquakes...) just fine. Question: what advantage could a specialized time-series DB have (how might we architect it to be better), compared to using an existing DB type?

Sample Answer:

- Time-series DB stores "append-only data". The data that arrives is almost always recorded as a new entry. The data typically arrives in time order. Time is a primary axis.
- Time-series DB can handle large scale data.
- Time-series DB typically include built-in functions and operations common to time-series data analysis.

Give 1.0 point if any reasonable advantage is provided. The answer might be more general/vague, BUT if it includes the fact that the **'time' column is handled as a special one** (eg.

the entries are stored in timestamp (sorted) order that makes it easy to do time-based queries), that's a good answer.

Q2. What is data governance? How might there be a bias, in ML classification results? How would these two intersect, ie. how can governance prevent or minimize bias in ML? Discuss the three questions, using an example for the third question.

Sample answer:

Data governance means setting internal standards—data policies—that apply to how data is gathered, stored, processed, and disposed of. This ensures the effective and efficient use of information.

[2 points if defined and explained properly]

[Give 1 mark if only advantages are explained]

Bias in ML classification results can occur due to multiple reasons like:

- Unbalanced/skewed training data set=> inherent bias which is picked up by the model
- Not enough data in the training sample which makes it hard for the model to learn hence resulting in bias.

[Any other valid reason other than above can also get full points].

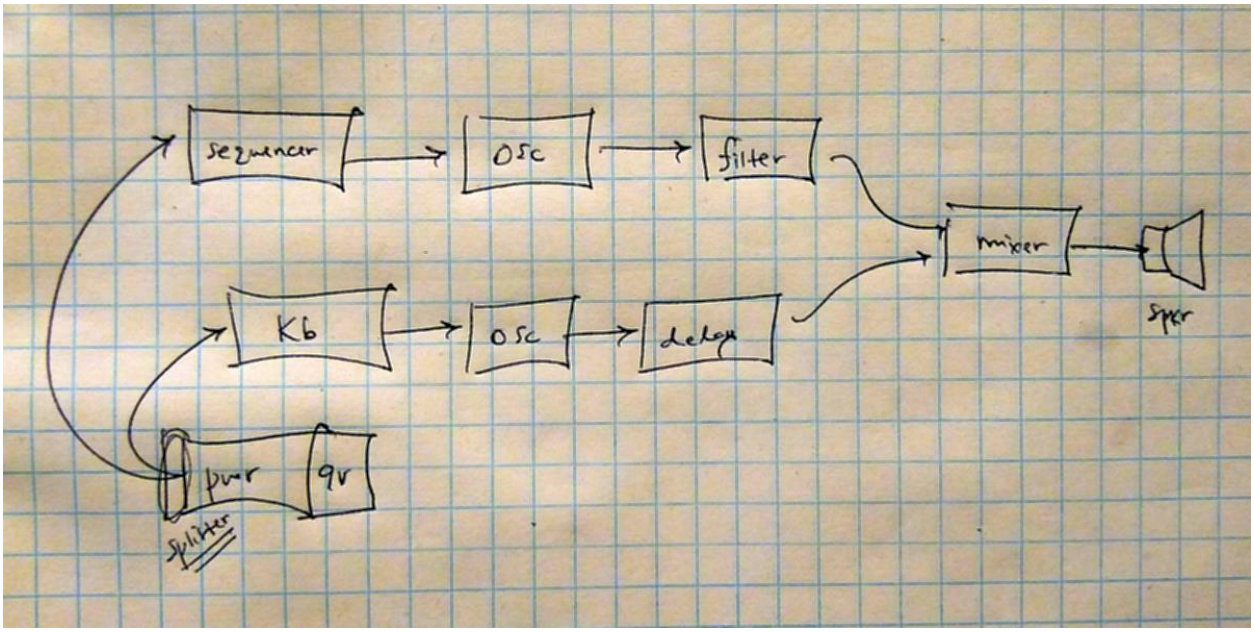
[Give 1 mark if only definition of Bias is explained]

Appropriate Governance policies in place, help in Curation's producing the required business data. For ex, an Organization might have a policy that collects sales data for a product in all seasons rather than just in the holiday festive season to have accurate sales prediction.

Governance+curation could include AUDITING data regularly, to examine BIAS and eliminate it by requesting more balanced data from the data originators.

[Any valid example with explanation gets full points]

Q3.



Shown above is a schematic, of the littleBits-based modular synth we looked it (and heard!). Each rectangle (except the bottom one) is a module that plays a part in creating/shaping sound.

A. How does the above, relate to data? In other words, data 'modules' can be loosely classified into 3 types (we did name them in class) - what are they?

Answer:
1. Data **generators** (eqvt to creating)
2. Data **modifiers**/filters/transformers (eqvt to changing)
3. Data **aggregators/combiners** (eqvt to mixing)

1 point each, for the three types above (any of the words mentioned is ok); an answer that says sources, transformers (filters) and sinks is ALSO ok!!

B. What is the architecture called, where we process data in stages? Name the architecture, and briefly discuss its advantage (over conventional data processing).

The architecture is called **dataflow.** Its advantage: unlike a script/program that needs to ALL be executed any time a PART of it changes, in dataflow, **ONLY downstream 'dirty' nodes that get affected by the change needs to be re-rerun** - this results in enormous processing cost savings.

1 point for mentioning dataflow, 1 point for explaining downstream processing (loose language is ok but MUST describe the 'only some nodes need to rerun' aspect).

Q4 1 mark for each direction. Deduct ½ marks if the difference is not mentioned clearly.

Sample answer:

1. **Higher (graph) level processing**, using Pig, Hive, etc., rather than explicitly using map()
and reduce()
2. **In-memory processing**, using Spark: specifically meant for iterative processing of data. It is considered an alternative to MapReduce
3. Flink: Similar to MapReduce but also **generalizes** it. Flink offers Map and Reduce functions but also additional transformations like Join, CoGroup, Filter, and Iterations.
4. Storm: reliable real time processing of unbounded **streams** of data
5. BSP: Just like MR but allows for parallel **nodes to also exchange data**, using communication and synchronization step.

Please note that there can be other directions also. All valid directions with correct explanations receive +1 each. Total 5 marks.

Q5

A. Variations:

. Star Schema: M:1 relationship between the fact table and each dimension table.

a. Snowflake Schema: Each dimension table can have its own dimension tables i.e, they can be normalized to create a 1:M chain.

b. Redundant fact tables: Create a separate fact table for each attribute in a dimension hierarchy.

c. Denormalize the fact table completely by adding extra dimension columns to it and filling them with redundant data. Speed vs Disk space expense tradeoff has to be considered.

[Any 3 variations from the above. 1 point per variation, Max 3]

A. Modern alternative to the traditional BI scheme aka Data Warehouse is **Data Lake**.

[1 point for naming the above schema (Data Lake)]

It is a better alternative because:

a. It offers a more continuous form of analytics.
b. Data is not ETL'ed but stored in its natural form
c. 'Schema on read' where we create a schema after storing raw data in a database

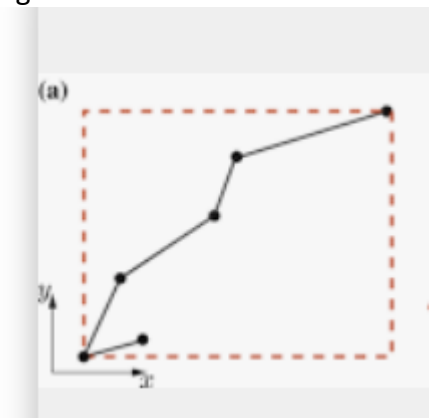
[1 point for reasoning why it is a better alternative]

Q6.

A. There is an inefficiency in the ‘R-tree’ scheme of indexing spatial data - what is it? You can illustrate with a diagram if you like.

Answer: the use of **AABB (Axis Aligned Bounding Boxes)** can be wasteful when features (eg road or building) is not parallel/perpendicular to X and Y.

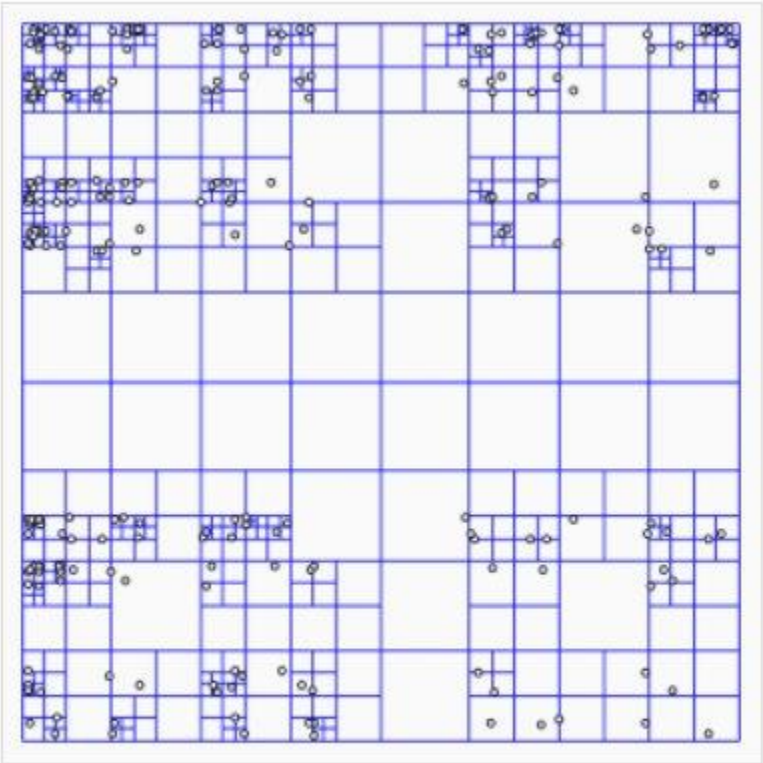
Eg:



1 point for explaining AABB (other wording than above is ok), but must talk about wasted space inside the bounding box.

B. How does a ‘quadtree’ help in indexing standard (eg long,lat) geospatial data? Briefly explain how it works.

Answer: From the slides:



A representation of how a quadtree divides an indexed area. Source: Wikipedia

The idea is to recursively split a containing square into 4 subsquares, but STOP recursing when a subsquare becomes empty, **forming a tree that has up to 4 splits at each level** (an empty square becomes an empty leaf node in the tree and stops getting split further). **To locate a feature, we simply go down one of the (up to 4) splits.**

1 point for explaining the above (alternate wording is ok) but the 'upto 4 splits' idea needs to be there (ie this is almost usually NOT a balanced tree at all!)

C. An 'octree' is a 3D extension of a quadtree (with 2^3=8 'cells', compared to 2^2=4 cells in quadtrees). What is an example of spatial data that octrees (rather than quadtrees) can help index?

Answer: an octree would contain **upto 8 splits** at each level (2^3=8), which is an extension of a quadtree that has up to 2^2=4 splits. We can **index features distributed in 3D space** using an

octree - eg. equipment in processing plants, merchandise in multi-floor departmental stores, even medical data (full body scan), sections of the night sky, etc.

1 point for any reasonable use (including ones not listed above - it MUST be for something that is in 3D (not 2D) space).

D. In HW3 you created a Spirograph curve overlaid on a map, that made use of (long,lat). Given (long,lat,height), we could create comparable structures in 3D, ie. spatially. What is an entertainment, and a commerce (ie to sell!) application of this "technology" (positioning in 3D), given drones with LED lights?

Entertainment application: **"fireworks"** (without the noise and chemicals!).

Commerce application: forming **logos** (eg Candy Crush did this recently in NYC with 500 drones), forming a QR code that people can scan using their phones for a product promotion or sale, etc.

1 point each for application - other applications that make sense are ok - there must be 2 different ones though.

Q7.

The different creation WAYS (NOT different chart TYPES mentioned above!!) include:

- coding using APIs such as matplotlib, D3…
- using dedicated software such as OriginPlot
- using math analysis software such as MATLAB or Mathematica
- using Excel
- using dataflow or point-and-click tools such as WEKA, RapidMiner…
- using cloud-based tools such as mode, Tableau, QLik etc
- using online (but not cloud) fill-in-with-data interfaces that create a chart png or pdf
- …

+1 (Upto 5) for each different kind of data visualization method examples mentioned (can be different from ones mentioned above as long as they are DISTINCT)

Q8. (3+1+1)
A. There exist MANY techniques for doing data mining (including ML). Name and discuss THREE different categories of them (they DO need to be distinct!).

Ans-
Different Categories of Data Mining Algorithms-

1. Classification: involves LABELING data
2. Clustering: involves GROUPING data, based on similarity
3. Association: involves RELATING data
4. Regression: involves COUPLING data

Can also be categorized into-

1. Supervised
2. Unsupervised
3. Semi-supervised
4. Self-supervised

The descriptions were mentioned in the lectures.

The answer can ALSO be this - the categories of techniques/tools might be

- **API-based** - eg keras, scikit-learn
- **point-and-click or dataflow based**, eg WEKA or KNIME
- **cloud-based** (upload data), run available steps/recipes (eg SiSense, Tableau etc)
- **math software based**, eg. ML libraries/functions in Mathematica or MATLAB

[Any 3 from the above. 1 point per variation, Max 3]

B. There is an unmistakable trend towards DIY (Do It Yourself) DM/ML. Specifically, what category of tools enables this?

Ans: Dataflow tools enable individuals to DIY DM/ML.

[1 point for mentioning the above]
[if not mentioned then give 0.3 point for mentioning any ONE specific tool covered in class]
[max 1 point for the question]

C. Also, what effect will the above (B.) have, on a traditional 'data scientist' role? Be Specific.

Ans: With tools enabling easy model creation and deployment, Data Scientist role will become obsolete. **The data scientist "standalone" role might become integrated into domain-specific USES, ie. the 'business ANALYST' role** is likely to be more valuable (where the analyst knows to use the tools AND knows about the data that gets passed through the tools).

[1 point for any reasoning similar to above]

Q9 (3+1+1 = 5 points)
A. Since the dawn of digital databases, there has been a need to 'connect' users to DBMS. How did such connectivity work, during the mainframe times, with PCs (specifically, Microsoft ones), and, coupled with web servers?

Ans: During the mainframe times, every relational database vendor had their own way of querying data known as Native SQL which was provided by vendors. There was a need to write a different code for each Database system with different syntax. Code for one Database could not be used for other databases. There was no standard of writing connectivity codes. Also, dumb terminals were used to connect to the mainframe DBs.

1 point for the above.

Microsoft used **ODBC Open Database Connectivity** (Superset of SQL) and added **DAO + Jet** for optimized interface, **RDO** for remotely accessing servers.
They also introduced **Object Linking and Embedding for Database (OLE-DB)** adding Object Oriented functionality for accessing data.
MSFT **ADO.NET** has critical features for development of distributed applications.
[1 points for mentioning any MSFT connectivity tools above]

Coupled with web servers: 'server-side scripts' (eg in Perl) were used - these ran on webservers, connected to DBs, fetched data, formatted results into HTML, sent it to the client.

1 point for the above.

B. How does modern connectivity work, given ubiquitous computing, storage, connectivity? Be specific!

Sample Ans: This is about MCC - Microservices + Containers + Cloud.

Applications are written to follow 'MCC' - tied together using loosely coupled microservices running on multiple clouds in multiple containers.

[1 points for any valid explanation of working of modern connectivity similar to above]

C. What advantage did SQL provide towards connectivity, when it came to multiple vendors implementing relational DBs (eg. Oracle, Microsoft, IBM etc.)?

Ans: SQL provided some level of **standardization**. It made things simpler as it was independent of the backend software used.
[1 points for any reasoning similar to above]

Q10.

A. There are many algorithms where iteration plays a key role in data mining. A few of them covered in the class are briefly explained below:

K-Means Clustering:

Start with 'n' random locations ('centroids') in the dataset. Assign each input point to the closest centroid. Compute new centroids based on the points in the cluster. Iterate till convergence is reached. By iterating over the data points and updating the centroids, we ensure the best clustering is reached.

AdaBoost:

AdaBoost is an ensemble learning technique where the weights of the base learner are iteratively and adaptively tweaked to minimize the overall classification error. [Starting from a larger number of features used by participating learners, the iterative training algorithm only selects only those features known to improve the predictive power of the overall model]

Expectation Maximization:

Start with random values for the model parameters. Compute probabilities for all latent variables using these parameters. Do a weighted average (by probability) to compute the best values of each latent variable (E-step). Use the latent variables from E-step to improve model parameters (M-Step). Iterate over E and M-step until the parameter values converge.

NNs too!! Backprop involves iterative modification of weights.

Full points [2 marks] if the answer briefly explains 2 DM algorithms that use iteration.
Deduct 0.5 mark for each algorithm that is not explained properly / incorrectly.
Deduct 1 mark if 2 relevant algorithm names are provided but not discussed.
Deduct 1 mark if only one relevant algorithm is provided and explained.

B. If we overfit the model on the input data, it will start to memorize the noise in the dataset and will **lose its ability to generalize** well to new data. If a model cannot generalize well to new data, then it will not be able to perform the classification or prediction tasks that it was intended for. (poor results on the testing dataset)

Full points [1 marks] if the answer talks about low testing accuracy/ loss of generalizability. OK if the answer talks about not learning the overall pattern (and instead, learning JUST the input data perfectly)

C. If the learning rate of the model to be very large, it will constantly overshoot the objective function and take large gradient steps. Due to this, the model can skip the optimal solution and might **not converge** at all.

Full points [1 marks] if the answer talks about suboptimal solution / overshooting the objective function / inability of the model to converge.

D. The sigmoid [or tanh] function at the end of each 'neuron' acts as an activation function and is used to apply **non-linearity** to the model. Since neurons are connected in layers, the non-linearities **GET COMPOUNDED**, which is why/how, ANY pattern in data can be learned, given a deep-enough network!

Full points [1 marks] if the answer talks about non-linearity, or compounding of it.