DS 5110: Introduction to Data Management and Processing (Fall 2024)

Sample Final Quiz

- You have 1.5 hours.
- The exam is closed book, closed notes.
- You are allowed to bring two letter-sized pages of notes (on both sides).
- Use of electronic devices is not allowed.
- Please read all instructions carefully.

Name: _____

NEU ID (optional): _____

1 Multiple-Choice Questions (50 points)

Circle **ALL the correct choices**: there may be more than one correct choice, but there is always at least one correct choice. **NO partial credit**: the set of all the correct answers must be checked. There are 10 multiple choice questions worth 5 points each.

1. When is a NoSQL database **not** a suitable replacement for a relational database?

   (a) When the data is unstructured and rapidly changing.

   (b) When the data cannot fit into the memory of a single machine.

   (c) When transactions are critically important for the application.

   (d) When dealing with big data applications that require high-speed data processing and analytics.

   (e) When strict schema enforcement is required for the application.

   (c)
   (a): False. NoSQL databases are suitable for unstructured and rapidly changing data structures.
   (b): False. NoSQL databases are designed to scale horizontally, allowing them to handle large datasets that cannot fit into the memory of a single machine.
   (c): True. When transactions are critically important, relational databases are preferred because they provide robust ACID compliance, which ensures data integrity and consistency.
   (d): False. NoSQL databases are commonly chosen for big data applications that require high-speed data processing and analytics.
   (e): True. NoSQL databases are flexible and do not enforce a strict schema, making them unsuitable when strict schema enforcement is required.

2. What are the advantages of using the BSON format over JSON?

   (a) BSON is optimized for data storage and retrieval.

   (b) BSON supports more data types than JSON.

   (c) BSON is more human readable than JSON.

   (d) BSON automatically compresses data to reduce storage space.

   (a) and (b).
   (a): True. BSON is a binary format that is optimized for data storage and retrieval, making it more efficient in terms of performance compared to JSON.
   (b): True. BSON supports a wider range of data types, including 'Date' and 'Binary', which are not available in JSON.
   (c): False. JSON is more human-readable because it is a text-based format, while BSON is a binary format and not easily human-readable.
   (e): False. BSON does not automatically compress data; any data compression must be implemented separately, usually as a feature of the database.

3. Suppose you have the following collection in MongoDB. The collection is named `contacts`.

```
{"_id" : 1, "region" : "NW1", "leads" : 1, "email" : "mlangley@co1.com"}
{"_id" : 2, "region" : "NW1", "leads" : 1, "email" : "jpicoult@co4.com"}
{"_id" : 3, "region" : "NW1", "leads" : 2, "email" : "zzz@company2.com"}
{"_id" : 4, "region" : "SE1", "leads" : 8, "email" : "mary@hssu.edu" }
```

```
{"_id" : 5, "region" : "SE2", "leads" : 4, "email" : "janet@col.edu"}
{"_id" : 6, "region" : "SE2", "leads" : 2, "email" : "bill@uni.edu"}
{"_id" : 7, "region" : "SE2", "leads" : 4, "email" : "iii@company1.com"}
{"_id" : 8, "region" : "SW1", "leads" : 1, "email" : "phil@co3.com"}
{"_id" : 9, "region" : "SW1", "leads" : 2, "email" : "thomas@company.com"}
{"_id" : 10, "region" : "SW2", "leads" : 2, "email" : "sjohnson@uchi.edu"}
{"_id" : 11, "region" : "SW2", "leads" : 5, "email" : "tsamuel@someco.com"}
```

How many documents will be returned from the following query?

```
db.contacts.aggregate([
    {"$group": {"_id": "$region", "count": {$count: {}}}},
    {"$match": {"count": {"$gte": 3}}}
])
```

(a) 0

(b) 1

(c) 2

(d) 3

(e) 4

(c). The first stage, `$group`, groups by the region and uses the `$sum` accumulator expression to count the number of documents in each group. Next, these documents flow into the `$match` stage, where documents with a count that is less than 3 (3 out of the 5 groups) are filtered out, returning two documents.

```
{ "_id" : "SE2", "count" : 3 }
{ "_id" : "NW1", "count" : 3 }
```

4. Suppose you have the following collection called `orders` in MongoDB:

```
[
  { "_id": 1, "customer": "John Doe", "items": [{ "product": "Laptop", "price
    ": 1200 }, { "product": "Mouse", "price": 50 }] },
  { "_id": 2, "customer": "Jane Smith", "items": [{ "product": "Laptop", "
    price": 1200 }, { "product": "Keyboard", "price": 100 }] },
  { "_id": 3, "customer": "Mike Johnson", "items": [{ "product": "Smartphone",
     "price": 800 }] },
  { "_id": 4, "customer": "Sarah Brown", "items": [{ "product": "Smartphone",
    "price": 800 }, { "product": "Laptop", "price": 1200 }] }
]
```

Which MongoDB query finds all orders where the total price of items in the order is greater than $1000?

(a) `db.orders.find({ "items.price": { $sum: { $gt: 1000 }}})`

(b) `db.orders.find({ "items": { $elemMatch: { "price": { $gt: 1000 }}}})`

(c)
```
db.orders.aggregate([
    { $unwind: "$items" },
    { $group: { _id: "$_id", totalPrice: { $sum: "$items.price" }}},
    { $match: { totalPrice: { $gt: 1000 }}}
])
```

(d)
```
db.orders.aggregate([
    { $project: { totalPrice: { $sum: "$items.price" }}},
    { $match: { totalPrice: { $gt: 1000 }}}
])
```

(c) and (d).
(a): False. `$sum` is an aggregation pipeline operator and cannot be used inside a `$find` query.
(b): False. This query looks for orders where at least one item in the items array has a price greater than $1000. It does not calculate the total price of all items in the order.
(c): True. This query first unwinds the items array, then groups the documents back by their `_id` while summing the prices of the items, and finally filters for those documents where the total price exceeds $1000.
(d): True. This query uses the `$project` stage to calculate the total price for each order by summing the prices in the items array, and then uses `$match` to filter for orders where the total price is greater than $1000. This is a more efficient approach compared to option 3 as it avoids the `$unwind` step.

5. In HDFS, what is the role of the NameNode?

   (a) Stores the Block IDs and locations of any given file in HDFS.

   (b) Executes file system namespace operations such as renaming files and directories.

   (c) Creates and deletes data blocks.

   (d) Chooses which DataNodes store the replicas of a given file.

(a), (b), (d)
(a): True. The NameNode stores the Block IDs and locations of files, maintaining the metadata for HDFS.
(b): True. The NameNode executes file system namespace operations, such as renaming files and directories.
(c): False. The NameNode does not handle the creation or deletion of data blocks; this is the responsibility of the DataNodes.
(d): True. The NameNode chooses which DataNodes store the replicas of a given file to ensure data reliability and redundancy.

6. Which of the following statements related to MapReduce are true?

   (a) Each mapper/reducer must generate the same number of output key/value pairs as it receives on the input.

   (b) The input to reducers is grouped by key.

   (c) The output type of keys generated by mappers must be of the same type as their input.

   (d) The output type of keys generated by mappers must be of the same input type of keys received by reducers.

(b), (d)
(a): False. Mappers and reducers can emit a different number of output key/value pairs compared to the number of input pairs they receive.
(b): True. The input to reducers is grouped by key, ensuring that all values for a particular key are processed together.
(c): False. The output key type of mappers does not have to match the input key type; they are defined independently.
(d): True. The output key type of mappers must match the input key type of reducers to ensure proper data flow between the map and reduce phases.

7. In Spark, which of the following operations triggers the execution of RDD transformations?

   (a) `map()`
   (b) `reduce()`
   (c) `transform()`
   (d) `select()`
   (e) `collect()`

   (b) and (e). These two functions are actions that trigger the execution of RDD transformations to bring the data from the RDD to the driver program.

8. What does the generalization error of a machine learning model represent?

   (a) The error rate of the model on the test data.
   (b) The variance of the model's predictions on the test data.
   (c) The expected value of the error on new input.
   (d) The difference in error between the training and testing data.

   (c).
   (a): False. While the generalization error influences the error rate on the test data, it specifically refers to the expected error on new, unseen data, not just the fixed test dataset.
   (b): False. The variance in the model's predictions reflects how much the model's predictions fluctuate for a given input but does not directly relate to the error.
   (c): True. The generalization error is indeed the expected value of the error on new input data. It quantifies how well a model is expected to perform when making predictions on data it hasn't been trained on.
   (d): False. The difference in error between training and testing data can indicate issues like overfitting or underfitting but does not define the generalization error, which is about how the model performs on any new input.

9. You apply standard scaling to the training set using its mean and variance. Which statement about standardizing the test set is true?

   (a) The test set should be standardized using its own mean and variance.
   (b) The test set should be standardized using the training set's mean and variance.
   (c) The test set should be standardized using the overall mean and variance of both the training and test sets combined.
   (d) The test set should not be modified in any way.

(b). When applying standard scaling, it is crucial to use the same parameters (mean and variance) for both the training and test sets to ensure consistency in the scale of the data. Using the training set's statistics for the test set prevents data leakage and ensures that the model is evaluated on the same scale as it was trained.

10. In Scikit-Learn, which of the following statements is true about the `Pipeline` class?

    (a) It sequentially applies a series of data transformations followed by a final estimator.

    (b) The last estimator in the pipeline must have a `predict` method.

    (c) It automatically optimizes hyperparameters of each step in the pipeline.

    (d) It helps to prevent data leakage by ensuring that data transformations are only learned from the training set.

(a), (d).
(a): True. `Pipeline` sequentially applies a series of transformations and ends with an estimator.
(b): True. The last estimator in the pipeline must have a `predict` method; otherwise, calling `pipeline.predict()` will result in an error.
(c): False. `Pipeline` does not automatically optimize hyperparameters; this must be done using `GridSearchCV` or a similar approach.
(d): True. Using a `Pipeline` helps prevent data leakage by ensuring that data transformations are only learned from the training set.

2 MongoDB (30 points)

You have a collection called `movies`. A sample document in the collection is shown below:

```
{'_id': ObjectId('573a1394f29313caabcdf639'),
 'title': 'Titanic',
 'released': datetime.datetime(1953, 7, 13, 0, 0),
 'plot': 'An unhappy married couple deal with their problems on board the ill-
    fated ship.',
 'genres': ['Drama', 'History', 'Romance'],
 'runtime': 98,
 'cast': ['Clifton Webb','Barbara Stanwyck','Robert Wagner','Audrey Dalton'],
 'directors': ['Jean Negulesco'],
 'writers': ['Charles Brackett', 'Walter Reisch', 'Richard L. Breen'],
 'imdb': {'rating': 7.3, 'votes': 4677},
 'reviews': [{'username': 'user1', 'comment': 'THIS MOVIE IS AMAZING!!! I love
    watching it!'},
    {'username': 'user2', 'comment': 'Knowing it is the 3rd highest grossing film
    of all time as of now, I can imagine why!'},
    {'username': 'user3', 'comment': 'This movie is a masterpiece, everything is
    perfectly and beautifully shot and well acted.'}]
}
```

Write the following queries in MongoDB (5 points for each query):

1. Find how many movies belong to the 'Romance' genre.

   ```
   db.movies.find({genres: 'Romance'}).count()
   ```

2. Find the titles and the ratings of the three movies with the highest IMDB rating.

   ```
   db.movies.find({}, {title: 1, 'imdb.rating': 1}).sort({'imdb.rating': -1}).
       limit(3)
   ```

3. Find the actor/actress who participated in the highest number of movies.

   ```
   db.movies.aggregate([
       {$unwind: '$cast'},
       {$group: {'_id': '$cast', 'num_movies': {$count: {}}}},
       {$sort: {'num_movies': -1}},
       {$limit: 1}
   ])
   ```

4. Find the average number of reviews written per movie. Note that some movies don't have any reviews.

   ```
   db.movies.aggregate([
       {$project: {num_reviews: {$size: {$ifNull: ['$reviews', []]}}}},
       {$group: {_id: null, reviewsAvg: {$avg: '$num_reviews'}}}
   ])
   ```

5. Add an actor named "Samuel L. Jackson" to the movie "Pulp Fiction".

   ```
   db.movies.updateOne(
       {'title': 'Pulp Fiction'},
       {$push: {'cast': 'Samuel L. Jackson'}}
   )
   ```

3 MapReduce (20 points)

For each of the following problems describe how would you solve it using MapReduce. Write the map and reduce tasks in pseudo-code.

1. The input is a list of housing data where each input record contains information about a single house: (address, city, state, zip, price). The output should be the average house price in each zip code.

```
function map(address, city, state, zip, price):
    emit(zip, price)

function reduce(zip, list of prices):
    emit(zip, avg(prices))
```

2. The input contains two lists. The first list provides voter information for every registered voter: (voter-id, name, age, zip). The second list gives occupancy information: (zip, age, job). For each unique pair of zip and age values, the output should give a list of names and a list of jobs for people in that zip code with that age. If a particular zip/age pair appears in one input list but not the other, then that zip/age pair can appear in the output with an empty list of names or jobs, or you can omit it from the output entirely.

   The mapper task first has to identify from which list the input record came from. In this case, we can use the number of fields in the record to identify the list. The output of the mapper will be key = (zip, age) and the value = either a name or a job with an additional label to distinguish which it is.

```
function map(record):
    if len(record) == 4:  # Voter information
        emit((record[3], record[2]), ('name', record[1])
    else:
        emit((record[0], record[1]), ('job', record[2])
```

   The reducer task will get all the names and jobs associated with a specific (zip, age) key and will aggregate the two lists together:

```
function reduce((zip, age), list of values):
    list_of_names = all values that are names from the voter list
    list_of_jobs = all values that are jobs from the occupancy list
    emit((zip, age), list of all names, list of all jobs)
```

   If a particular age/zip pair appears in one input list but not the other, it will naturally result in an empty list for either names or jobs.