Spesifikasi Tugas Seleksi Asisten Lab AI ‘23

Dipersiapkan Oleh Tim Asisten Lab *Artificial Intelligence* ‘22
Versi: 1.0 07/08/2025

Deadline: 5 September 2025 23.59 WIB

Spesifikasi

1. AI Theory

Jawablah pertanyaan-pertanyaan berikut:

1. Apa yang dimaksud dengan AI (*Artificial Intelligence*)?
2. Jelaskan 4 jenis AI menurut Russel dan Norvig!
3. Apa yang dimaksud dengan *Agent* pada AI?
4. Jelaskan konsep PEAS yang digunakan untuk memperjelas suatu lingkungan tugas (*task environment*) pada suatu permasalahan AI!
5. Jelaskan 6 properti dari suatu lingkungan tugas!
6. Jelaskan minimal 5 jenis *agent* pada AI! Urutkan mulai dari yang paling sederhana hingga yang paling kompleks.
7. Carilah **satu topik permasalahan** yang sekiranya dapat diselesaikan **menggunakan AI**. Kemudian elaborasikan jenis AI yang digunakan, deskripsi masing-masing komponen pada PEAS, klasifikasi 6 properti lingkungan tugas, serta jenis agent yang digunakan.

2. Design of Experiment + Supervised Learning

Salah satu jenis *agent* yang paling sering digunakan saat ini adalah *Learning Agent*. Pada tugas seleksi ini, Anda akan diminta untuk membangun suatu ***learning agent* yang dapat menyelesaikan masalah yang telah anda sebut pada Bagian 1**. *Learning agent* yang akan dibuat pada bagian ini termasuk ke dalam teknik *supervised learning*, dimana *agent* akan menggunakan data yang sudah berlabel untuk memprediksi nilai target.

Data Gathering

Carilah dataset yang dapat digunakan untuk melatih model yang akan anda gunakan untuk menyelesaikan permasalahan. Dataset yang digunakan dibatasi pada jenis *structured data* saja. Kemudian, tentukanlah fitur target yang sesuai dengan permasalahan yang akan diselesaikan. Sumber data dibebaskan kepada peserta seleksi. Beberapa platform yang menyediakan open datasets adalah sebagai berikut: Kaggle, UCI Machine Learning Repository, awesomedata/awesome-public-datasets: A topic-centric list of HQ open datasets. (github.com).

Exploratory Data Analysis

Exploratory Data Analysis merupakan tahap untuk memahami data secara lebih lanjut dan merangkum karakteristik dari data yang dimiliki. Lakukanlah **minimal** beberapa hal ini untuk tahap EDA:

- Statistik dasar data tiap kolom/fitur
    - Fitur numerik: jumlah data, rata-rata, standar deviasi, nilai minimum, nilai maksimum, persentil ke-25, persentil ke-50, persentil ke-75
    - Fitur kategorikal: jumlah jenis data (*unique values count*), jumlah data per *unique value*
- Persebaran data untuk setiap kolom/fitur
- Pengecekan outlier
- etc. (Tambahkan jika perlu, terutama untuk eksplorasi hal-hal yang berkaitan dengan *domain* data yang dimiliki)

Data Preprocessing

Data Preprocessing merupakan tahap pengolahan data sedemikian sehingga data sudah dalam format atau kondisi yang tepat untuk digunakan ke tahap selanjutnya. Beberapa proses yang biasa dilakukan di tahap *data preprocessing* adalah sebagai berikut:

- Data cleaning (Handling missing values, remove duplicates, correct error and inconsistencies)
- Data transformation (normalization, standardization, categorical data encoding)
- Feature selection
- Dimensionality reduction

- Balancing target feature (undersampling, oversampling)
- etc.

Untuk tahap ini, lakukanlah proses-proses yang sekiranya diperlukan sesuai dengan data yang dimiliki dan model yang akan digunakan. Untuk setiap proses, **sertakanlah alasan mengapa proses tersebut perlu untuk dilakukan** (alasan ditulis di markdown cell pada notebook saja).

Modeling and Validation

Modeling merupakan tahap perancangan algoritma atau model statistik untuk membantu dalam menganalisis data lebih lanjut atau untuk melakukan prediksi berdasarkan data yang dimiliki. Sementara *validation* adalah tahap evaluasi kinerja dari model yang telah dirancang. Berikut merupakan algoritma yang wajib Anda implementasikan:

- KNN (*K-Nearest-Neighbors*)
    - **Minimal** bisa menerima 2 input parameter
        - **Jumlah tetangga**
        - **Metrik jarak** antar *data point*. Minimal dapat menerima 3 pilihan, yaitu Euclidean, Manhattan, dan Minkowski
- Logistic Regression/Softmax Regression/Polynomial Regression
    - Implementasikan algoritma yang sesuai dengan permasalahan yang Anda selesaikan
        - Logistic Regression untuk binary classification
        - Softmax Regression untuk multiclass classification
        - Polynomial Regression untuk regression
    - Gunakan ***gradient descent*** untuk optimasi parameter model
        - Loss function yang digunakan adalah ***mean squared error*** (Polynomial Regression) atau ***cross-entropy*** (Logistic/Softmax Regression). Boleh menambahkan pilihan lain sebagai *loss function*, asalkan sesuai dengan task yang dipilih. Penambahan implementasi akan dihitung sebagai **bonus**.
        - Menerima masukan input ***learning rate,*** **jumlah iterasi,** ***regularization term*** (antara l1, l2, atau tidak menggunakan *regularization term*), dan ***loss function*** (jika mengimplementasikan lebih dari 1 loss function)

    - Implementasi polynomial regression dapat menerima input orde polinomial yang akan digunakan
    - **Bonus:** Selain *gradient descent*, implementasikan *Newton's Method* sebagai alternatif untuk optimasi parameter model
- Gaussian Naive Bayes (hanya untuk yang memilih **task klasifikasi** saja)
- CART (*Classification and Regression Tree*)
    - Sesuaikan pembuatan algoritma berdasarkan permasalahan yang diselesaikan (klasifikasi/regresi)
- SVM (*Support Vector Machine*)
    - Cara optimasi parameter model **dibebaskan** (boleh gradient descent, quadratic programming, ataupun metode lain)
        - Input parameter disesuaikan dengan pilihan optimizer yang akan diimplementasikan
    - SVM yang diimplementasikan harus **mampu menangani data yang *not linearly separable*.**
    - **Minimal** mengimplementasikan kernel jenis linear, tambahan implementasi kernel lain dianggap sebagai **bonus**.
- ANN (*Artificial Neural Network*)
    - Implementasikan model ANN jenis Feedforward Neural Network yang terdiri dari 1 Input Layer, n Hidden Layer $(n \geq 0)$, dan 1 Output Layer. Seluruh *layer* termasuk ke jenis *Fully Connected Layer*.
    - ANN yang diimplementasikan terdiri dari 3 tahap kalkulasi per *epoch*, yaitu:
        - *Forward propagation*
        - *Backward propagation*
        - *Weight update*
    - Model yang diimplementasikan minimal dapat:
        - Menerima **fungsi aktivasi *sigmoid*, ReLU, *linear*, dan *softmax***. Tambahan implementasi selain dari yang disebutkan akan dianggap sebagai **bonus**. Pengaturan fungsi aktivasi berada pada tingkat *layer*.
        - Menerima metode inisialisasi
        - Menerima **fungsi loss berupa *mean squared error* dan *cross-entropy (log loss)*.** Tambahkan pilihan untuk menambahkan *regularization term* atau tidak (antara l1, l2, atau tidak ada). Tambahan implementasi selain dari yang disebutkan akan dianggap sebagai **bonus**.

        - Dioptimasi dengan metode *batch gradient descent.*
            - Menerima masukan input ***learning rate,*** **jumlah iterasi (*epoch*),** **dan *batch size***.
            - Tambahan metode optimizer yang lain (misal Adagrad, Adam, dll) akan dianggap sebagai **bonus**. Sesuaikan parameter input dengan optimizer yang dipilih.
        - Menerima input jumlah *neuron* tiap *layer*.
        - **Bonus:** Gunakan konsep *automatic differentiation* dalam perhitungan gradien pada ANN.
    - Sangat disarankan untuk memperhatikan *modularity* dalam implementasi ANN. Sebagai contoh, Anda dapat mengambil TensorFlow atau PyTorch sebagai referensi.

Kemudian untuk validasi, berikut beberapa hal yang Anda harus lakukan:

- Pilihlah *metric* untuk mengevaluasi kinerja dari model yang telah diimplementasikan. Kemudian, **jelaskanlah mengapa *metric* yang Anda pilih sudah sesuai dengan permasalahan yang Anda selesaikan** (alasan dituliskan pada markdown cell pada notebook saja).
- Gunakan strategi **hold-out validation** dan **k-fold cross-validation** untuk mengevaluasi model. Pastikan tidak ada *data leakage* antar *data split* pada proses *validation*.
- Jawablah pertanyaan-pertanyaan berikut (simpan jawaban Anda dalam file modeling.pdf):
    a. Jelaskan apa yang dimaksud dengan *hold-out validation* dan *k-fold cross-validation*!
    b. Jelaskan kondisi yang membuat *hold-out validation* lebih baik dibandingkan dengan *k-fold cross-validation*, dan jelaskan pula kasus sebaliknya!
    c. Apa yang dimaksud dengan *data leakage*?
    d. Bagaimana dampak *data leakage* terhadap kinerja dari model?
    e. Berikanlah solusi untuk mengatasi permasalahan *data leakage*!

Untuk setiap model diatas, lakukanlah hal-hal berikut:

1. Jelaskan cara kerja dari algoritma tersebut! (boleh dalam bentuk *pseudocode* ataupun narasi)

2. Implementasikan algoritma tersebut *from scratch* dengan bahasa pemrograman Python. Anda hanya diperbolehkan untuk menggunakan *library* untuk perhitungan matematika (NumPy, Pandas, Scipy, dan sejenisnya).
3. Implementasikan algoritma tersebut menggunakan salah satu *library machine learning* (scikit-learn, Pytorch, Tensorflow, dan sejenisnya).
4. Bandingkan hasil evaluasi model pada nomor 2 dan 3, bagaimana hasil perbandingannya? Jika ada perbedaan, jelaskan alasannya!
5. Jelaskan *improvement* apa saja yang bisa Anda lakukan untuk mencapai hasil yang lebih baik dibandingkan dengan hasil yang Anda punya saat ini! *Improvement* yang dimaksud tidak terbatas pada bagaimana algoritma diimplementasikan, namun juga mencakup tahap sebelum *modeling and validation*.

Untuk nomor 1, 4, dan 5, simpan jawaban Anda dalam file [Nama-Algoritma].pdf.

3. Unsupervised Learning

Berbeda dengan *supervised learning*, *unsupervised learning* menerima input berupa data yang tidak berlabel (atau tidak menggunakan label) untuk menemukan pola atau *insight* pada data tersebut. Pada bagian ini, Anda diharuskan untuk mengimplementasikan model berikut:

- K-means
    - Menerima input *jumlah cluster* dan *maximum iteration*
    - Metode inisialisasi *cluster* yang digunakan adalah *random initialization*.
    - **Bonus:** Tambahkan implementasi metode inisialisasi *k-means++*, kemudian jadikan metode inisialisasi sebagai parameter input
- DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*)
    - Menerima input epsilon, *minimum samples*, dan *distance metric* (*Euclidean*, *Manhattan*, dan *Minkowski)*
- PCA (*Principal Component Analysis*)
    - Menerima input *jumlah komponen*
    - Selain output data hasil PCA, model **minimal** harus menyimpan persentase *explained variance* dari tiap komponen

Untuk setiap model di atas, lakukanlah hal-hal berikut:

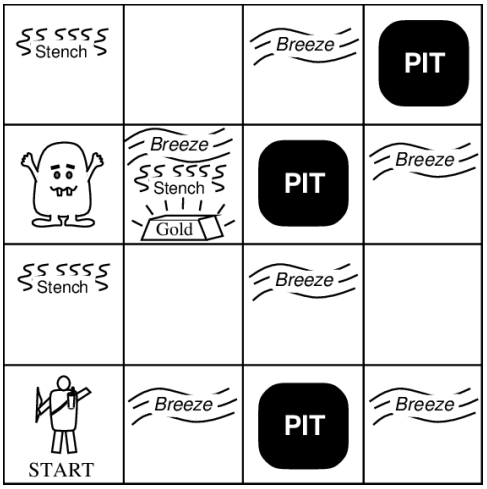
1. Jelaskan cara kerja dari algoritma tersebut! (boleh dalam bentuk *pseudocode* ataupun narasi)

2. Implementasikan algoritma tersebut from scratch dengan bahasa pemrograman Python. Anda hanya diperbolehkan untuk menggunakan *library* untuk perhitungan matematika (NumPy, Pandas, Scipy, dan sejenisnya).
3. Implementasikan algoritma tersebut menggunakan library *scikit-learn*.
4. Bandingkan hasil yang didapatkan pada nomor 2 dan 3, bagaimana hasil perbandingannya? Jika ada perbedaan, jelaskan alasannya!

Untuk nomor 1 dan 4, simpan jawaban Anda dalam file [Nama-Algoritma].pdf.

Untuk bagian ini, Anda diperbolehkan untuk menggunakan dataset lain (tambahkan file Jupyter Notebook baru jika memakai dataset yang berbeda) yang bukan terkait dengan permasalahan pada Bagian 2. Anda juga diperbolehkan untuk menggunakan implementasi Anda pada bagian ini untuk mendukung analisis Anda pada Bagian 2 (misalkan melakukan *dimensionality reduction* dengan PCA yang sudah Anda buat).

4. Reinforcement Learning



Reinforcement learning merupakan salah satu jenis *machine learning* dimana suatu *agent* melakukan aksi terhadap lingkungannya, kemudian *agent* akan menerima *reward* atau *penalty* terhadap aksi yang telah dilakukannya. Tujuan akhir dari interaksi tersebut adalah membuat *agent* berperilaku sedemikian sehingga dapat memaksimalkan *reward* yang diterima. Pada bagian ini, Anda akan diminta untuk mengimplementasikan spesifikasi berikut:

- Terdapat suatu *game* sederhana dengan spesifikasi sebagai berikut:

    - Lingkungan (Environment)
        - Sebuah grid berukuran **4x4**.
        - Posisi **Start** berada di kotak **[1,1]** (pojok kiri bawah).
        - Lokasi **Wumpus**, **Gold**, dan **Pit** sesuai dengan gambar di atas.
            - **Wumpus**: **[1, 3]** (Stench di sekitarnya).
            - **Gold**: **[2,3]** (Glitter di kotak yang sama).
            - **Pits**: **[3, 1]**, **[3, 3]**, **[4, 4]** (Breeze di sekitarnya).
        - Agen dapat merasakan **Stench** (bau), **Breeze** (angin), dan **Glitter** (kilauan) sesuai aturan standar Wumpus World
    - Aturan Permainan dan Poin (*Rewards*):
        - Setiap aksi yang dilakukan (bergerak, berbalik) dikenai penalti **-1 poin**.
        - Jika agen masuk ke kotak berisi **Wumpus** atau **Pit**, permainan berakhir dan agen mendapat penalti **-1000 poin**.
        - Jika agen berhasil mengambil **Gold** (dengan aksi **Grab** di kotak yang benar), agen mendapat imbalan **+1000 poin**.
        - Permainan dianggap **sukses (menang)** jika agen berhasil mengambil Gold dan kembali ke kotak **[1, 1]** lalu melakukan aksi **Climb**.
- Implementasikan permainan Wumpus World sesuai dengan spesifikasi di atas dengan ketentuan:
    - Implementasikan algoritma **Q-Learning** dan **SARSA** *from scratch* (hanya boleh menggunakan library untuk perhitungan matematis seperti NumPy/Pandas).
    - Agen harus belajar sebuah kebijakan (policy) untuk menavigasi lingkungan, mengambil emas, dan kembali ke titik awal.
    - Tampilkan **Q-table final** yang dihasilkan setelah proses *learning* selesai untuk kedua algoritma.
    - Visualisasikan atau tampilkan **jalur (path) optimal** yang ditemukan oleh agen dari titik awal menuju emas dan kembali lagi, berdasarkan Q-table yang telah dipelajari.
    - Bahasa pemrograman yang digunakan bebas. Tampilan simulasi juga dibebaskan (boleh berbasis CLI maupun GUI).
- Jawablah pertanyaan berikut (simpan jawaban Anda dalam file reinforcement-learning.pdf):
    - Jelaskan cara kerja dari algoritma **Q-Learning** dan **SARSA**, terutama perbedaan fundamental antara keduanya (*on-policy* vs *off-policy*).

- Bandingkan hasil dari kedua algoritma tersebut dalam konteks Wumpus World ini. Analisis perbandingan bisa mencakup:
  - Kecepatan konvergensi (jumlah episode yang dibutuhkan untuk belajar).
  - Kebijakan (policy) final yang dihasilkan. Apakah ada perbedaan?
  - Jalur (path) yang ditempuh. Apakah Q-Learning cenderung mengambil rute yang lebih berisiko dibandingkan SARSA? Jelaskan mengapa!

BONUS

1. Ensemble Methods

Tambahkan implementasi model yang termasuk ke dalam **kategori *ensemble* pada Bagian 2**. Jumlah model tambahan yang diimplementasikan dibebaskan.

2. Convolutional Neural Networks

Untuk bonus 2, Anda akan menambahkan implementasi **2 jenis layer tambahan** selain Dense/Fully Connected layer pada implementasi ANN yang telah anda buat pada **Bagian 2**, yaitu:

- 2D Convolution Layer
  - Menerima input jumlah filter konvolusi
  - Menerima input *kernel size*
  - Menerima input *stride length* untuk masing-masing dimensi
  - Menerima input *use_padding*. Jika dilakukan padding, input dan output data akan memiliki dimensi yang sama.
  - Menerima input jenis fungsi aktivasi (sama seperti FC layer pada Bagian 2)
- 2D Max Pooling Layer
  - Menerima input *pool size*
  - Menerima input *stride length* untuk masing-masing dimensi

Setelah anda mengimplementasikan 2 layer yang baru (*from scratch*), lakukanlah hal-hal berikut:

- Rancanglah minimal 2 dari 6 arsitektur model sebagai berikut: LeNet, AlexNet, VGGNet, GoogLeNet, ResNet, ZFNet
- Latihlah model yang anda buat dengan dataset MNIST, kemudian lakukan tahap validasi untuk mengecek kinerja model.
- Implementasikan 2 arsitektur model yang Anda buat sebelumnya menggunakan *library deep learning*, kemudian bandingkan hasilnya dengan implementasi model *from scratch*. Jika ada perbedaan, jelaskan alasannya!
- Jawablah pertanyaan berikut (tambahkan jawaban Anda ke dalam file ANN.pdf):
    - Jelaskan apa yang dimaksud dengan convolution layer dan max pooling layer!
    - Jelaskan mengapa CNN relatif memiliki kinerja yang lebih baik dibandingkan FC layer untuk kasus klasifikasi gambar!

3. Text Classification

Lakukanlah pengembangan model klasifikasi teks (*emotion classification*) secara end-to-end (mulai dari *EDA* hingga *validation*) dengan sumber data sebagai berikut: meisaputri21/Indonesian-Twitter-Emotion-Dataset: Indonesian twitter dataset for emotion classification task (github.com)

Catatan

- Utamakan spesifikasi WAJIB dibandingkan dengan bonus
- *By default*, penggunaan *library* apapun **dibebaskan**, kecuali untuk beberapa bagian yang diperintahkan untuk mengimplementasikan model *from scratch*.
- **Berikut beberapa referensi yang mungkin dapat membantu:**
    - Artificial Intelligence - A Modern Approach, 3rd Ed. - Russell, Norvig.pdf
    - ml-road/resources/Hands On Machine Learning with Scikit Learn and TensorFlow.pdf at master · yanshengjia/ml-road (github.com)
    - scikit-learn: machine learning in Python — scikit-learn 1.5.1 documentation
    - Tutorials | TensorFlow Core
    - Learn the Basics — PyTorch Tutorials 2.4.0+cu121 documentation
    - Hugging Face - Learn
    - Learn Python, Data Viz, Pandas & More | Tutorials | Kaggle
    - Kaggle Notebooks

- 16. Learning: Support Vector Machines (youtube.com)
- The spelled-out intro to neural networks and backpropagation: building micrograd (youtube.com)
- Reinforcement Learning: An Introduction With Python Examples | DataCamp
- Q-Learning Explained - A Reinforcement Learning Technique (youtube.com)
- Q-Learning vs. SARSA | Baeldung on Computer Science
- Wumpus World in AI

FAQ

Pertanyaan dapat ditanyakan pada link **FAQ** berikut. Pastikan pertanyaan yang ditanyakan tidak berulang.

Aturan

- Tidak diperbolehkan melakukan **copy paste** tanpa memahami code dari sumber manapun!
- Seleksi dikerjakan secara **mandiri** oleh masing-masing peserta seleksi.

Deliverables

1. Kumpulkan link drive/repository ke form berikut.
2. Bagian 1 (AI Theory) dikumpulkan dalam bentuk link Google Drive dengan format pdf.
3. Bagian 2 (DoE + Supervised Learning) dan bagian 3 (Unsupervised Learning) dikumpulkan dalam bentuk ***link* satu *repository* github**. Sedangkan bagian 4 (Reinforcement Learning), bonus 3 (Text Classification) masing-masing dikumpulkan dalam bentuk ***link repository* github yang berbeda**. Masukkan juga jawaban setiap pertanyaan dalam format pdf. Contoh struktur *repository* untuk bagian 2 dan 3:

```
nama-repository
├── dataset
│   ├── dataset1.csv
│   └── dataset2.csv
│
├── src
│   ├── supervised-learning
│   │   ├── knn.py
│   │   └── logistic-regression.py
│   │
│   ├── unsupervised-learning
│   │   ├── dbscan.py
│   │   └── pca.py
│   │
│   └── doe.ipynb
│
├── answers
│   ├── supervised-learning
│   │   ├── modeling.pdf
│   │   ├── knn.pdf
│   │   └── logistic-regression.pdf
│   │
│   └── unsupervised-learning
│       ├── dbscan.pdf
│       └── pca.pdf
│
└── README.md
```

4. Setiap *repository* **wajib** berisi README.md yang berisi deskripsi singkat penggunaan program. Tambahkan *checklist* dan daftar bonus yang dikerjakan pada *repository* bagian 2 (DoE + Supervised Learning), bagian 3 (Unsupervised Learning), dan bagian 4 (Reinforcement Learning) berisi algoritma yang diimplementasikan. Bonus 1 dan 2 termasuk ke dalam implementasi Bagian 2, sehingga masukkan ke dalam daftar bonus bagian 2 jika mengerjakan. Untuk Bonus 3 dan 4 terpisah dari repository spek wajib, sehingga tidak perlu menambahkan di daftar dalam readme. Contoh:

```
Supervised Learning (Bagian 2)
[ ] KNN
[ ] LogReg/SoftReg/PolyReg (tulis yang dikerjain aja)
[v] Gaussian Naive Bayes (kalo tasknya regresi, gausah ceklis tp kasih
keterangan)
[v] CART
[ ] SVM
[v] ANN
Bonus yang diimplementasikan:
   - … (silakan list sendiri)
   - …

Unsupervised Learning (Bagian 3)
[ ] K-MEANS
[v] DBSCAN
[v] PCA
Bonus yang diimplementasikan:
   - … (silakan list sendiri)
   - …

Reinforcement Learning (Bagian 4)
[ ] Q-LEARNING
[v] SARSA
```

5. Pengumpulan dilakukan dengan membuat release dengan tag vX pada repository yang telah Anda buat sebelum deadline, dengan X adalah angka revisi dimulai dari 0 jika tidak ada revisi/versi awal.