AI Engineer position

Technical assessment of Arthur Vieira

Document designated to guide the assessment of the technical skills for a candidate from different perspectives for the role of AI Engineer within SDG Group’s Data Technologies practice.

Strategy. Decision. Governance.

AI Engineer role technical assessment for SDG Group's Data Technologies practice.

Aim of this technical assessment is for the candidate **to demonstrate needed skills regarding Python development, data science lifecycle basic concept understanding and surrounding engineering** key aspects of production ready advanced analytics projects.

You might not know all components listed below for the deployment of an end-to-end solution, but it is also a key aspect of the proof to test your ability to have a basic understanding of some of those and been able to use them within the context of this challenge in a coherent manner. These are the key aspects to showcase during the oral demonstration:

1. If you are not familiar to Docker, is a free accessible container managing technology that will help you deploy most of the rest of the technologies required for this challenge. Being a Windows user, you need to be aware of the version required for Docker Desktop to work, which in general is Windows 10 on its version where, from our experience WSL2 backend is mostly recommended. Docker Desktop is also available for Mac users as for Linux users it is more usual to have it directly install by means of their package managing software.
2. Use Docker to deploy an **Apache Spark** local instance. You can use this DockerHub image, following the instruction to enable *two local workers*:
    1. Download deployment specification as a Docker Compose template (for example using cURL):

curl -LO https://raw.githubusercontent.com/bitnami/bitnami-docker-spark/master/docker-compose.yml

    2. From the path where the template was downloaded you can just call it to be deployed using the command prompt and following command: *docker-compose up*
    3. From there, by means of your browser you could just access Apache Spark's UI interface in the following direction: http://localhost:8080/.
3. Setup a **HDFS file system** as well by means of docker compose- You can find an example here.
4. From **this Databricks' blog article**, download the notebook with the example code here with the objective of transforming the original code in a production grade PySpark application (usually invoked using spark-submit command). This is a key aspect given that one of key conditions for an AI Engineer role is not only being able to understand such monolithic code but also to refactor its different parts in separated modules so that each part can be used separately in an automated model creation pipeline where model gets trained and scored in different points of the model lifecycle and the model created and used in those phases gets usually stored as a separated structure to the code generating it (a Pickle file often). As you will see the *notebook* starts by creating a data table containing relevant features so that it can afterwards be used in both phases, training and scoring of the model, taking into account each model corresponds to a different group. This distributed action is done by means of Spark so the data gets spread on different worker nodes applying an User Defined Function (UDF) to each subset. This last part is the one we require you to re-structure and industrialize eliminating all experimentation and visualization parts more relevant during exploratory phases. While building the PySpark application you will face different problems related to the self-service nature of the Databricks environment, related to environment specifics or dependency management that you will need to solve.
5. You will need to also **deploy an Apache Airflow** instance (you might use Docker for that as well). Apache Airflow is a process orchestrator that differs from other solutions as the orchestration processes are

defined using Python scripts. You could use following guidelines to deploy Airflow using docker compose as before.

6. Once you have Airflow running and you have familiarized with it, the way dags are created and executed we **would like you to create your own DAG** where the tipical time series pipeline Will be implemented. By using your Apache Spark deployment, your HDFS file system and your Apache Airflow instance using appropriate operator for the work (SparkSubmitOperator, for example). Your DAG will execute following process:
    1. Upload required CSV files to the HDFS
    2. Execute the ML steps (train, test, score…) that you have created on step 4
    3. Download the models stored inside HDFS (some at least) to your local environment

The aim of this exercise is no other than showing you understand the usual lifecycle when comes to machine learning model deployment in productive environments. Batch predictions where also periodic or upon request model training phases are performed are part of the routine in many organizations today. Obviously, an enterprise solution would require this to happen automatically, with a set of tools that allows both monitoring and managing the whole lifecycle of these processes.

Once you have achieved this, we invite you to add some extra features to your solution (you could also use Docker to ease their deployment):

- MLflow local instance to track experiments performed, store trained models or serve them to scoring steps as part of an MLOps pipeline.
- You could also use Flask or FastAPI libraries in order to expose one of those trained models as an online request service. You can encapsulate all this within a Docker image to ease its deployment and scalability afterwards.
- Also, it would be valued any addition to the solution regarding monitoring (tools such as Prometheus or Grafana, for example).

Of course, all these extra points are additional to the initial requirements and can be tackled according to your expertise or availability. Apart from the technical implementation, it also will be valued if you have conducted your own research on tools, solutions or architectures that could be presented on a presentation to support your vision on the solution delivered apart from the points above stated. If you feel comfortable on adding visualizations or automating infrastructure deployment, feel free to do so, it will be very much valued.

If you have doubts at any time during the development of the assessment, feel free to contact and ask for clarifications. It also applies to any issue that may come up requiring to move the date. Feel free to reach out to Joan (joan.marchan@sdggroup.com - 669285434) or Jesús (jesusvicente.garcia@sdggroup.com - 669727656).

Thanks so much in advance for the effort and hope to talk soon. Please do inform us that you have received the technical assessment and that you accept the challenge.

SDG Group

W. www.sdggroup.com
M. info.es@sdggroup.com