**REPORT**

**Question 1:** Describe how this application can be deployed in a cloud environment.

In general, creating a machine learning model is becoming simpler, quicker, and more available to all. The major cloud providers, such as Microsoft's Azure Machine Learning, Amazon's AWS SageMaker, and Google Cloud AI Platform, all have dedicated services for designing machine learning solutions. By implementing these services, you can handle the entire machine learning lifecycle from start to finish, by implementing MLOps (machine learning-oriented DevOps) to create, train, and deploy models in automated pipelines They all support well-known frameworks and programming languages (such as Python, R, scikit-learn, XGBoost, Tensorflow, or PyTorch), making the transition from working locally as painless as possible.

With the knowledge of the usage of cloud services, it can be easier for the programmers to understand the tasks in a more convenient way by facilitating the communication and thus makes the productization of the machine learning solution fasters.

Firstly, to load the application on cloud, we need to store it by using the picking method. This model basically converts the model into binary form that can be stored, copied, moved, transferred, and loaded in order to retrieve the original one. Here, in Python, we will use the “joblib” library that enables an efficient storage for the Python objects that may contain large data structures.

For the filename, joblib or .bin are examples of supported extensions. It is also possible to use other extensions (such as. z) for further compression, if needed.

Along with that, to deserialize the model and ensure all compatibilities, python versions and dependencies are required for Azure Machine Learning and to indicate scikit-learn’s version.

**Steps to Deploy an application on Azure Machine Cloud**

* **Installing the Azure MLSDK:** The Azure Machine Learning portal GUI can be used to deploy an already qualified model, but to deploy a web endpoint in a single container (which is the fastest way to deploy a model), command-line method can be used. In our local terminal, run the following command to instal the Azure ML SDK with the default packages:

**Pip install –upgrade azureml -sdk**

* **Creating a workspace on Azure:** We'll build a new resource group because we're starting from scratch on this project. We can choose the Basic workspace version since it includes all the features we need. We can access the workspace via the Azure Machine Learning dashboard once the resource group and workspace have been developed. We press Change subscription > Download config file, in the Azure ML portal's upper-right corner navigation bar and place the file in our project directory. You can also get this config file by going to the newly developed workspace in Azure and clicking Download config file on the Overview tab. We can deploy our model directly from our Python IDE once we have this file in our project folder.
* **Registering a model:** To transfer and enroll our all-around prepared model, we go to the Models area on the left-side menu and click on Register model. We name our model, show the system we used to prepare it (for this situation, scikit-learn), and alternatively compose a portrayal for extra metadata. We peruse our machine to choose and transfer our serialized model record, and afterward we click on Register.

This entire methodology can likewise be immediately finished with a Python content and the Azure ML SDK. We first need to interface with the workspace utilizing the from\_config() technique, which will look for our config document inside the root index of the venture. At that point, we can enroll our prepared model by utilizing the Model class. After running this content, we can return to the Azure Machine Learning gateway, and check that our model has been enlisted under the Models area.

* **Deploying the model:** Once our model is enrolled, we would already be able to send it with no guarantees. This implies that our conveyed web administration will accept a similar info information as our prepared model (for this situation, a variety of four gliding numbers signifying our four chose includes), and will yield the expectation to do as such, we find our enlisted model in the workspace, and utilize the send () strategy from the Model class, giving a name to our web administration and the pointer to the model. Thusly, our model will be conveyed in an ACI (Azure Container Instance). ACIs in Azure Machine Learning are intended to be utilized for advancement and testing purposes, as they just help low-scale responsibilities on CPUs.
* **Alternative deployment with custom entry script:** If this register target arrangement doesn't give sufficient computational capacity to our model? For sure in the event that we utilize certain libraries that are excluded from the default climate. Along these lines, to have greater adaptability to characterize how the web administration interfaces with outer clients: for example, we may need to do extra tasks like normalizing the information highlights or managing missing qualities in some specific manner with two stages, for example, determining a derivation design (by giving a passage script to our web administration and a custom environment) and a sending arrangement.

**Question 2:** Describe how you would implement security features for your app in the cloud.

* **Know what you are responsible for:** All cloud administrations are not something similar, and the degree of obligation changes. Programming as-a-administration (SaaS) suppliers ensure their applications are secured and that the information is being communicated and put away safely, yet that is not generally the situation with IaaS conditions.
* **Access Control:** Use the Identity and access control tools provided by all the clouds. Know who has access to what data and when. When creating identity and access control policies, award the base arrangement of advantages required and incidentally award extra authorizations depending on the situation. Arrange security gatherings to have the narrowest r possible; use reference security bunch IDs where conceivable. Consider apparatuses, for example, CloudKnox that let you set admittance controls dependent on client action information.
* **Data Protection:** Use the encryption tools and the management services as Encryption is a safeguard — regardless of whether a security arrangement fizzles and the information falls under the control of an unapproved party, the information can't be utilized.
* **Secure the accreditations:** Make sure to routinely change the keys, to try not to give attackers time to block traded off keys and penetrate cloud conditions as advantaged clients. Try not to utilize the root client account, not in any event, for authoritative errands. Utilize the root client to make new client with assigned privileges. Lock down the root account (maybe by adding multifactor validation [MFA]) and use it just for explicit record and administration the board errands.
* **Improve visibility:** Major cloud providers all offer some degree of logging tools, so try to turn on security logging and observing to see unapproved access endeavors and different issues. It can likewise be utilized for change tracking, resource management, security analysis and consistence reviews.
* **Adopt a shift-left approach to security:** The shift-left development advocates consolidating security contemplations ahead of schedule into the improvement interaction as opposed to adding security in the last phases of advancement. "In addition to the fact that undertakings should screen what they have in IaaS stages, they ought to check all their code that is going into the stage before it goes live," says McAfee's Flaherty. "With shift-left, you're reviewing for and getting expected misconfigurations before they become an issue." Look for security tools that incorporate with Jenkins, Kubernetes and others to mechanize the inspecting and adjustment measure.

**Question 3:** If you need to make your application serverless how it can be done.

* **Static Web Hosting:** It is the pivotal step in building the serverless application. Amazon S3 is equipped for facilitating the static web assets which incorporate HTML, JavaScript, CSS and the picture records which are stacked in the program.
* **User Management:** Amazon Cognito can give the client management and to validate capacities for getting the backend API.
* **Serverless Backend:** The DynamoDB from Amazon gives a tireless layer. It is the place where the information can get stored by the utilization of API's capacity by the use of the AWS Lambda.
* **Restful APIs:** The JavaScript that gets executed in the program has the capacity of sending and accepting information from public.
* **Resource Cleanup:** The creation of the resources all through the tutorial will get ended.