1. **Train the model on an ongoing basis, as new data becomes available?**

Establishing an automated training pipeline that can often trigger the arrival of new data. The Pipeline will then be integrated with data sources for the data ingestion and preprocessing of the new data. Finally, by making sure the training process is repeatable for consistency. The workflow must automate data ingestion, data cleaning and featurization of the data. Version control can be used for the code and to track changes over time.

**2. Ensure the data coming in is of good quality?**

Because of "garbage in, garbage out," you must keep an eye on the data your model is consuming upstream in the pipeline. Your model reacts differently depending on the input data it gets. Model performance will drastically suffer if the statistical distribution of the trained data differs from that of the production data.

That Quality can be ensured by the following measures:

* Data cleaning: The data should be checked for missing data, Outliers, and any inconsistencies
* Data drift: Data drift is another problem with data quality to be aware of. This only indicates that the statistical characteristics of the data have changed.
* Data Governance: Defining the clear ownership of data.
* Quality Metrics; Key metrics like completeness, accuracy, and consistency should be used to measure the quality of data.

1. **Monitor model's performance?**

**Monitor performance metrics**: Depending on the kind of problem (classification, regression, etc.), periodically assess the model's performance using relevant measures like Accuracy, precision, nDCG, recall, F1-score, or Mean squared error.

**Keep an eye out for idea drift**: Look for modifications in the way the goal variable and input features relate to one another. Model drift can be identified with the aid of methods such as classifier-based approaches, statistical tests, and hypothesis testing. Concept drift can also result from model drift.

**Compare against a benchmark**: To evaluate the performance of the current model, use a benchmark model, such as a more straightforward or stable model. Model drift may be indicated if the performance of the main model starts to deteriorate in comparison to the benchmark.

**4.0 Make the model available to other services as part of a webapp?**

It takes careful planning to deploy a machine learning model and make it accessible as a component of a web application, taking user experience, scalability, and integration into account. The following outlines can be followed to deployed a machine learning app:

* **Model Serialization**: The best trained model is save d into a pickle or joblib file format suitable for deployment.
* **Web framework selection**: appropriate web framework should be chosen based on the team expertise and the designed need.
* **API Design and Development:**
* RESTfulAPI or GraphQL endpoint can be used to expose the machine learning model. By defining the input parameters, authentication, and response formats.
* **Containerization:** Dockers can be used to containerize the web app and its dependencies.
* **Orchestration**: Scalability concerns can be addressed by using a container orchestration tool like Kubernetes.
* **User Interface Integration:**
* Building a front end for the users to interface with the app. HML, CSS, React.js and Javascript can be used for the front end and back-end design of the web app.
* **Security Measures:** This can be ensured by implementing authentication and authorization mechanisms. Implementing API Key authentication or OAuth for secure access.
* **Monitoring and Logging:** It is important to monitor for performance, errors, and usage patterns. Therefore, it is necessary to implement logging to capture relevant information for troubleshooting. Tools like Prometheus or Gafana can be used in this case.
* **Continuous Integration/ Continuous Deployment:** This can be used for automated testing and deployment. It helps for consistent updates to the web without having any downtime. GitLab CI, Github Actions AND Jenkins can be used for this purpose.
* **Documentation:** Provide clear and comprehensive documentation for developers and users of the application.

Tools that can used for this:

* TensorFlow Extended (TFX) for data validation
* Jenkins and GitLab For CI/CD
* Prometheus and Grafana for Monitoring and alerting
* MLflow and Data Version Control (DVC) for model versioning and Experiment tracking
* Google AutoML, Azure AutoML, H2O.ai for automated machine learning modeling
* Docker and Kubernetes for containerization and orchestration
* HashiCorp Vault for security compliance
* Cloud services: AWS SageMaker, Google AI Platform, Azure machine Learning

performance will drastically suffer if the statistical distribution of the trained data differs from that of the production data.