Python Environment Setup

Required Software and Versions:

- Python Version: Use Python 3.9 , as it provides better support for type hints and newer library features.

- Integrated Development Environment (IDE): Recommend using PyCharm or Visual Studio Code with Python extensions for code editing and debugging.

- Package Manager: Use `pip` for installing Python libraries and `conda` (ANACONDA) for managing environments.

Installation Steps:

Install Python:

- Download and install Python 3.9 or higher https://www.python.org/downloads/ .

- Python and Pip are added to the system's PATH.

- I want to use Python 3.9 personally because more string methods such as str.removeprefix() and str.removesuffix(), which can simplify string manipulation.

- Helpful while dealing with API's.

Install Anaconda :

- Download Anaconda from the https://www.anaconda.com/products/distribution.

- Follow the installation instructions to set up Anaconda, which includes `conda`.

- WHY Anaconda ?- It comes with pre installed datascience and ML libraries- such as NumPy, Pandas, Scikit-learn,TensorFlow, and PyTorch.

- Conda environments can be exported and shared among all developers.

- Anaconda works on Windows, macOS, and Linux.

ALTERNATIVES- DOCKER, MINICONDA.

Setup a Virtual Environment:

- Using conda: `conda create --name jerichoai_env python=3.9`
- Activate the environment: `conda activate jerichoai_env`

-**INSTALLING PACKAGES:**

- conda install numpy pandas scikit-learn matplotlib
- pip install tensorflow

- **Managing environments :**

-conda env list

-conda deactivate

-conda remove --name jerichoai_env --all

Install Essential Python Libraries:

-ACTIVATE PYTHON 3.9:

conda activate jerichoai_env

-Install data manipulation and computation libraries:

conda install numpy pandas

-Install M.L libraries:

conda install scikit-learn

pipinstall tensorflow==2.6.0

conda install pytorch torchvision torchaudio cudatoolkit=10.2 -c pytorch

pip install pytorch-lightning

-Install Visualisation Tools:

conda install matplotlib seaborn

- Install Jupyter Notebook: conda install jupyter

- Cloud SDKs: pip install boto3 azure-sdk google-cloud-storage.

-Managing and Updating Libraries: pip install requests sqlalchemy

AI/ML Platform Development Setup

- **Machine Learning Libraries**: TensorFlow (2.6.0 for compatibility with newer features and stability), PyTorch, Scikit-Learn.

TENSOR FLOW in JerichoAI: Can be used for predictive analytics as forecasting cloud costs and usage patterns, anomaly detection in cloud spend.

PyTorch in JerichoAI: Best for developing deep learning models that require flexibility and speed during experimentation, such as real-time anomaly detection in cloud operations.

Scikit-Learn in JerichoAI: Useful for less complex but critical tasks as regression analysis, clustering, and classification, often needed for initial data analysis and baseline model building.

- **Data Manipulation Tools**: Pandas, NumPy.

Pandas in JerichoAI: Key for preprocessing steps such as cleaning data, transforming features, merging datasets, and handling time-series data from cloud usage statistics.

NumPy in JerichoAI: Critical for any operation that requires high-speed mathematical computations such as transformations, simulations, and algorithm implementations.

- **Visualization Tools**: Matplotlib, Seaborn.

Matplotlib in JerichoAI: Can be used to visualize cost trends over time, the distribution of spending across different services or departments, and outliers in data.

Seaborn in JerichoAI: Useful for creating more complex visualizations like heatmaps of cloud service usage or pair plots of multiple features affecting cloud costs.

- **Model Training and Evaluation**: PyTorch Lightning (simplifies training workflows).

PyTorch in JerichoAI: rapid development and training of models by managing the training loop, allowing developers to focus more on the actual modeling and less on the training mechanics.

Development Steps:

Data Handling:

- Use `pandas` and `numpy` for data manipulation and preprocessing.

- Integrate with cloud data sources using `boto3` for AWS, `azure-sdk` for Azure, and `google-cloud-storage` for Google Cloud.

Model Development:

- Develop cost prediction and anomaly detection models using `TensorFlow` or `PyTorch`.

- Utilize `scikit-learn` for simpler ML models and metrics.

Model Training and Evaluation:

- Use `PyTorch Lightning` to simplify the training process.

- Setup experiments to log training sessions with tools like MLflow or TensorBoard.

API Development for Integration:

- Develop APIs using Flask or FastAPI to serve the ML model predictions.

- Ensure APIs are capable of handling requests to fetch model outputs and receive new data for predictions.

Version Control:

- Use Git for version control. Set up a repository on GitHub or GitLab to manage code versions and collaborate.

Continuous Integration/Continuous Deployment (CI/CD):

- Implement CI/CD pipelines using Jenkins, GitHub Actions, or GitLab CI to automate testing and deployment of the machine learning models.

Testing and Documentation:

- Write unit tests using `pytest` or `unittest`.

- Document the API and model usage using tools like Swagger or Redoc.

Deployment and Monitoring

- Deploy APIs on cloud platforms using Docker and Kubernetes for scalability and management.
- Monitor the model's performance and system health using Prometheus and Grafana.