Course Title: Python for Machine Learning and Deep Learning with TensorFlow

Duration : 10 days

Target Audience: Beginners to Intermediate professionals (e.g., Data Analysts, Software Engineers, Product Managers) looking to gain practical skills in ML and DL.

Learning Outcomes: By the end of this course, participants will be able to:

- Master essential Python libraries for data manipulation and visualization.
- Understand and implement core Machine Learning algorithms using scikit-learn.
- Grasp fundamental Deep Learning concepts and architectures.
- Build, train, and tune neural networks using TensorFlow and Keras.
- Apply ML techniques for anomaly detection and predictive maintenance.
- Evaluate and optimize model performance using various techniques.
- Understand the unique characteristics and challenges of time series data.
- Implement classical time series forecasting methods (ARIMA, SARIMA, SARIMAX).
- Leverage Deep Learning models (LSTM) for complex time series predictions

---

Module 1: Python for AI/ML & Environment Setup

- **Theory:**
    - Python Essentials for ML: NumPy for numerical operations, Pandas for data manipulation, Matplotlib and Seaborn for visualization.
    - Data Structures: Efficient handling of data with lists, dictionaries, and Pandas DataFrames.
    - Working with Large Datasets: Strategies for memory management and efficient processing.
    - Environment Setup: Jupyter Notebooks and Integrated Development Environments (IDEs) for ML development.
    - Overview of the Open-Source ML Ecosystem: Introduction to key libraries and tools.
- **Hands-on: Data Manipulation and Exploratory Visualization**
    - **Exercise:** Load, clean, and perform exploratory data analysis (EDA) on a dataset to extract initial insights.
    - **Tools/Libraries:** Python, NumPy, Pandas, Matplotlib, Seaborn
    - **Dataset:** Open Source Dataset
    - **Expected Outcome:** A cleaned dataset ready for modeling and initial data visualizations illustrating key trends and relationships.

Module 2: Introduction to Machine Learning

- **Theory:**
    - Types of Machine Learning: Supervised, Unsupervised.
    - Problem Framing: Understanding and categorizing ML problems into Regression, Classification, and Clustering.
    - Machine Learning Workflow and Pipeline: A step-by-step approach from data to deployment.
- **Hands-on: Building Your First Classification Model**
    - **Exercise:** Implement a basic classification model using scikit-learn on a well-known dataset.
    - **Tools/Libraries:** Python, Scikit-learn
    - **Dataset:** Open Source Dataset
    - **Expected Outcome:** A trained classification model with an initial report on its accuracy.

Module 3: Supervised Learning in Depth

- **Theory:**
    - Regression Algorithms: Linear Regression, Ridge, and Lasso for predictive modeling.
    - Feature Preprocessing: Techniques for feature scaling, encoding categorical features, handling missing values, and outlier detection.
    - Classification Algorithms: Logistic Regression, Decision Trees, and Random Forests for categorical predictions.
    - Evaluation Metrics: Understanding and applying metrics like RMSE, Accuracy, Precision, Recall, and F1-score for model assessment.
- **Hands-on: Comparing Supervised Learning Algorithms**
    - **Exercise:** Apply and compare the performance of multiple supervised learning algorithms on a regression task.
    - **Tools/Libraries:** Python, Scikit-learn
    - **Dataset:** Open Source Dataset
    - **Expected Outcome:** A comparative table of model performance metrics and identification of the best-performing model for the given dataset.

Module 4: Unsupervised Learning & Feature Engineering

- **Theory:**
    - Clustering Techniques: K-Means and Hierarchical Clustering for data segmentation.
    - Dimensionality Reduction: Principal Component Analysis (PCA) for feature extraction.
- **Hands-on: Customer Segmentation using Clustering**
    - **Exercise:** Apply clustering algorithms to segment customers based on their attributes and visualize the results.
    - **Tools/Libraries:** Python, Scikit-learn, Matplotlib, Seaborn
    - **Dataset:** Open Source Dataset
    - **Expected Outcome:** Visualizations of customer clusters with an interpretation of the business insights derived from the segmentation.

Module 5: Anomaly Detection and Predictive Maintenance

- **Theory:**
    - Anomaly Detection Fundamentals: Identifying unusual patterns and outliers in data.
    - Common Anomaly Detection Techniques: Isolation Forest, One-Class SVM.
    - Predictive Maintenance Concepts: Using ML to forecast equipment failures and schedule maintenance.
    - Time Series Analysis for Predictive Maintenance: Features for time-series data.
- **Hands-on: Implementing Anomaly Detection and Predictive Maintenance**
    - **Exercise:** Apply anomaly detection technique on Open Source Dataset
    - **Tools/Libraries:** Python, Scikit-learn, Pandas
    - **Expected Outcome:** Identification of anomalous data points and a trained model that predicts potential failures or unusual activity patterns based on sensor data.

Module 6: Introduction to Deep Learning with TensorFlow

- **Theory:**
    - Deep Learning vs. Traditional Machine Learning: Understanding the fundamental differences and advantages.
    - Neural Network Basics: Concepts of neurons, layers, and activation functions (e.g., ReLU, Sigmoid).
    - Introduction to TensorFlow and Keras: Core functionalities, tensor operations, and building neural networks with Keras API.

    - Building Your First Artificial Neural Network (ANN): A foundational model for supervised learning.

- **Hands-on: Building and Training a Simple ANN**
    - **Exercise:** Construct and train a basic Artificial Neural Network for a classification task.
    - **Tools/Libraries:** Python, TensorFlow, Keras
    - **Dataset:** MNIST Handwritten Digits Dataset http://yann.lecun.com/exdb/mnist/
    - **Expected Outcome:** A trained ANN model achieving accuracy greater than 90% on the MNIST dataset.

Module 7: Deep Learning Architectures & Model Tuning

- **Theory:**
    - Convolutional Neural Networks (CNNs): Understanding convolution, pooling, filters, and their application in image processing.
    - Recurrent Neural Networks (RNNs): Exploring sequence modeling capabilities with LSTMs and GRUs.
    - Overfitting and Regularization: Techniques to prevent model overfitting.
    - Dropout and Batch Normalization: Methods for improving generalization and training stability.
    - Optimizers and Learning Rate Scheduling: Understanding common optimization algorithms and learning rate adjustment.
    - Hyperparameter Tuning: Techniques for finding optimal model parameters.
- **Hands-on: Image Classification and Model Optimization**
    - **Exercise:** Build and train a CNN model for image classification and optimize it using hyperparameter tuning and different optimizers.
    - **Tools/Libraries:** Python, TensorFlow, Keras, Scikit-learn (for hyperparameter tuning utilities)
    - **Dataset:** CIFAR-10 Dataset https://www.cs.toronto.edu/~kriz/cifar.html

Expected Outcome: An optimized CNN model trained on CIFAR-10 with improved accuracy and better generalization.

Module 8: Classical Time Series Methods (ARIMA, SARIMA, SARIMAX)

- **Theory:**
    - **Introduction to Time Series Data:** Understanding the nature of sequential data.

    - **Time Series Structure:** Identifying and analyzing components like trend, seasonality, and noise.
    - **Stationarity:** Its importance in time series analysis and methods to achieve it.
    - **Autocorrelation:** Understanding Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots for model identification.
    - **ARIMA Models:**
        - Autoregressive (AR) terms and their role.
        - Moving Average (MA) terms and their function.
        - Differencing (d) for stationarity.
        - Selecting appropriate p, d, q parameters.
    - **SARIMA Models:**
        - Incorporating seasonal components (P, D, Q, m).
        - Handling seasonal patterns in data.
    - **SARIMAX Models:**
        - Adding exogenous variables to enhance forecasting.
        - When to prefer SARIMAX over SARIMA.
- **Hands-on Lab: Forecasting with ARIMA, SARIMA, and SARIMAX**
    - **Objective:** Build and compare ARIMA, SARIMA, and SARIMAX models to forecast monthly airline passenger numbers.
    - **Tools/Libraries:** statsmodels, pandas, matplotlib, seaborn
    - **Dataset:** Open Source Dataset
    - **Expected Outcome:**
        - Generated forecasts for future months.
        - A comparison of model performance using metrics like RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) for ARIMA, SARIMA, and SARIMAX.

Module 9: Machine Learning & Deep Learning for Time Series (XGBoost, LSTM)

- **Theory:**
    - **Feature Engineering for Time Series:**
        - Creating lag features to capture past values.
        - Using rolling statistics (mean, std dev) for aggregated historical information.
    - **Time Series Train-Test Split:** Strategies for splitting data to avoid look-ahead bias.

- **XGBoost for Time Series:**
    - Understanding why gradient boosting models are effective for forecasting.
    - Reshaping time series data into a supervised learning format (features and target).
- **LSTM for Time Series:**
    - Recurrent Neural Networks (RNNs) vs. Long Short-Term Memory (LSTM) networks.
    - Handling long-term dependencies in sequential data.
    - Designing appropriate LSTM network architectures for forecasting.
    - Data normalization and sequence preparation for LSTM input.

- **Hands-on Lab 1 (XGBoost): Predict Daily Energy Consumption**
    - **Objective:** Predict daily energy consumption using lag features and an XGBoost model.
    - **Tools/Libraries:** xgboost, pandas, numpy, matplotlib
    - **Dataset:** Open Source Dataset
    - **Expected Outcome:**
        - A built and evaluated XGBoost model for energy consumption forecasting.
        - A plot comparing the forecast values against the actual observed values.
- **Hands-on Lab 2 (LSTM): Predict Stock Prices**
    - **Objective:** Predict stock prices using an LSTM network.
    - **Tools/Libraries:** tensorflow/keras, pandas, numpy, matplotlib
    - **Dataset:** Open Source Dataset
    - **Expected Outcome:**
        - A trained LSTM model on historical stock closing prices.
        - A visualization comparing the predicted stock prices against the actual historical prices.
