# Implementation of a Machine Learning Model with TensorFlow or Scikit-learn

**Objective:**

The objective of this project is to design and develop a Machine Learning model using either **TensorFlow** or **Scikit-learn**. This project will cover the entire Machine Learning development cycle, from data preprocessing to model training and evaluation.

**Project Steps:**

**1. Data Loading:**

- Use **Pandas** to load the data, display it, and perform an initial exploration of key features.

**2. Data Preparation:**

- Data cleaning: handle missing values, normalize numerical variables, and encode categorical variables (if necessary) using **Pandas** and **NumPy**.
- Split the dataset into training and testing sets to ensure proper model evaluation.
- Visualize variable distributions using **Matplotlib**.

**3. Data Processing:**

- Manipulate the data using **NumPy** and **Pandas** to prepare it for Machine Learning models.
- Separate features from labels, and, if needed, transform the data for model compatibility.

**4. Model Architecture Setup:**

**With TensorFlow:**

- Create an appropriate neural network model (regression or classification) using **TensorFlow** (or **Keras**, its high-level API).
- Configure the network layers (dense, convolutional, etc.) according to the requirements, and optimize hyperparameters such as the number of neurons, learning rate, and activation functions.

**With Scikit-learn:**

- Choose an appropriate model (logistic regression, SVM, KNN, etc.) and fine-tune its parameters (regularization and hyperparameters).
- Experiment with several baseline models for future comparison (decision tree, logistic regression, etc.).

**5. Model Training:**

**TensorFlow:**

- Train the model on the training set and display loss and accuracy curves for each epoch.
- Add cross-validation to fine-tune hyperparameters and monitor overfitting.
- Use regularization techniques (dropout, L2) and optimization methods (Adam, RMSprop).

**Scikit-learn:**

- Train the selected model on the training set using techniques like cross-validation to adjust parameters.
- Experiment with dimensionality reduction techniques (PCA, feature selection) to optimize performance.

**6. Model Evaluation:**

**Evaluation Metrics:**

- Calculate performance metrics such as **accuracy**, **recall**, **F1-score**, or **AUC** (depending on the problem).
- Visualize results using a **confusion matrix** for classification problems or a **ROC curve**.

**Performance Comparison:**

- If multiple models are tested, compare their respective performances using metrics like accuracy, F1-score, or RMSE.
- Visualize these results with **Matplotlib**.

**Model Improvement:**

- If necessary, further adjust hyperparameters or experiment with more complex models to enhance performance.

**Deliverables:**

- Well-documented and commented source code.
- A report explaining technical choices and presenting the obtained results.
- Graphs and visualizations of model performance, including loss curves, confusion matrices, or other relevant results.
- Critical analysis of the performance with suggestions for improvement.