**1-Day Machine Learning with Python - Bootcamp**

### 📅 Agenda Overview

| Time Slot | Topic |
| --- | --- |
| 9:00–9:30 AM | Introduction to ML & Python Ecosystem |
| 9:30–10:30 AM | Supervised Learning - Concepts & Hands-on |
| 10:30–11:30 AM | Unsupervised Learning - Concepts & Hands-on |
| 11:30 AM–12:00 PM | Semi-Supervised & Self-Supervised Learning |
| 12:00–1:00 PM | **Lunch Break** |
| 1:00–2:00 PM | Model Evaluation, Overfitting, and Regularization |
| 2:00–3:30 PM | Real-Life Projects (Classification + Clustering) |
| 3:30–4:30 PM | Advanced ML Concepts: Ensemble, Dimensionality Reduction |
| 4:30–5:30 PM | Modern ML Trends & Tools |
| 5:30–6:00 PM | Q&A + Next Steps |

### 🔧 Platform for Hands-On

* **Google Colab** - <https://colab.research.google.com>
* Alternative: **Jupyter via Anaconda** - <https://www.anaconda.com>

### 🧑‍🏫 Session 1: Introduction to Machine Learning (9:00–9:30 AM)

* What is ML? AI vs ML vs DL
* Types of Learning: Supervised, Unsupervised, Semi-Supervised
* Libraries: numpy, pandas, sklearn, matplotlib, seaborn

**Sample Code:**

import pandas as pd  
import seaborn as sns  
df = sns.load\_dataset('iris')  
df.head()

**Examples:**

* House price prediction
* Spam email classification

### 🔍 Session 2: Supervised Learning (9:30–10:30 AM)

* Regression vs Classification
* Algorithms: Linear Regression, Logistic Regression, Decision Trees

**Exercise: House Price Prediction**

from sklearn.linear\_model import LinearRegression  
from sklearn.datasets import fetch\_california\_housing  
from sklearn.model\_selection import train\_test\_split  
  
X, y = fetch\_california\_housing(return\_X\_y=True)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y)  
model = LinearRegression()  
model.fit(X\_train, y\_train)  
print("Prediction:", model.predict(X\_test[:1]))

### 🧩 Session 3: Unsupervised Learning (10:30–11:30 AM)

* No Labels
* Algorithms: K-Means, PCA

**Exercise: Customer Clustering**

from sklearn.cluster import KMeans  
from sklearn.datasets import make\_blobs  
import matplotlib.pyplot as plt  
  
X, \_ = make\_blobs(n\_samples=300, centers=4, cluster\_std=0.6)  
model = KMeans(n\_clusters=4)  
model.fit(X)  
plt.scatter(X[:, 0], X[:, 1], c=model.labels\_)  
plt.show()

### 🤖 Session 4: Semi-Supervised Learning (11:30–12:00 PM)

* Few labeled, many unlabeled
* Examples: Label propagation, used in NLP, images

### 🧪 Session 5: Model Evaluation (1:00–2:00 PM)

* Metrics: Accuracy, Precision, Recall, F1
* Overfitting, Cross-validation

**Code:**

from sklearn.metrics import accuracy\_score  
y\_pred = model.predict(X\_test)  
print("Accuracy:", accuracy\_score(y\_test, y\_pred))

### 💼 Session 6: Mini Projects (2:00–3:30 PM)

**Project 1: Iris Classification**

* Use: seaborn.load\_dataset('iris')

**Project 2: Customer Segmentation**

* Use: make\_blobs or Mall Customer dataset

### 🧠 Session 7: Advanced ML (3:30–4:30 PM)

* Random Forest, XGBoost
* PCA, t-SNE for dimension reduction

**Example:**

from sklearn.ensemble import RandomForestClassifier  
model = RandomForestClassifier()  
model.fit(X\_train, y\_train)

### 🔬 Session 8: Modern ML Trends & Tools (4:30–5:30 PM)

| Tool | Purpose |
| --- | --- |
| AutoML | Google AutoML, AWS SageMaker |
| HuggingFace | NLP models and transformers |
| PyCaret | Rapid ML experimentation |
| Gradio | Web UI for ML apps |
| Streamlit | Interactive ML dashboards |
| ONNX | Model interoperability |

**Example: AutoML with PyCaret**

!pip install pycaret  
from pycaret.datasets import get\_data  
from pycaret.classification import \*  
  
data = get\_data('diabetes')  
setup(data, target='Class variable')  
compare\_models()

### ✅ Final Exercises

1. Diabetes prediction from dataset
2. Build a Streamlit salary predictor
3. Use HuggingFace for sentiment classification

### 📚 Resources

* Book: *Hands-On ML with Scikit-Learn & TensorFlow* by Aurélien Géron
* Course: [Google ML Crash Course](https://developers.google.com/machine-learning/crash-course)
* Platforms: Kaggle, HuggingFace, Google Colab
* Continued Tools: DVC, Streamlit, AutoML, HuggingFace