Machine Learning with Python

Contents

1 Overview

Machine learning is a subset of AI that uses computer algorithms that require feature engineering. It teaches computers to learn from data and identify patterns and use them to make decisions without receiving explicit input from the user. There are three main types of machine learning models:

1. Supervised learning models are trained on a known set of features and a target variable, to identify relationships which are then used for inference or forecasting on new data. For example: linear regression.
2. Unsupervised learning models do not admit labelled feature-target style data, instead they are trained on a set of variables which are all considered features; the model finds relationships between these features. For example: Principal Component Analysis.
3. Reinforcement learning models simulate an AI agent interacting with its environment, they learn how to make decisions based on feedback from the environment. For example:

The two focuses of supervised learning are regression and classification One of the main types of unsupervised learning are clustering

1.1 Machine Learning Lifecycle

1. **Problem Definition:** Clearly state the objective and desired outcome.
2. **Data Collection:** Identify required data and its source.
3. **Data Preparation:** Clean the data, handle missing values, normalize if necessary, engineer features, and perform exploratory data analysis. Split into training and testing sets.
4. **Model Development:** Train the model, tune hyperparameters, and evaluate performance using appropriate metrics.
5. **Deployment:** Integrate the trained model into a production environment.

1.2 Difference between a Data Scientist and AI Engineer

Aspect	Data Science	AI Engineering
Primary Use Cases	Descriptive and predictive analytics (e.g., EDA, clustering, regression, classification)	Prescriptive and generative AI (e.g., optimisation, recommendation systems, intelligent assistants)
Data Type Focus	Primarily structured/tabular data, cleaned and preprocessed	Primarily unstructured data (text, images, audio, video), used at large scale
Model Characteristics	Narrow-scope ML models, smaller in size, domain-specific, faster to train	Foundation models, large-scale, general-purpose, high compute and data requirements
Development Process	Data-driven model development (feature engineering, training, validation)	Application of pre-trained models with prompt engineering, PEFT, and RAG frameworks

Table 1: Key Differences Between Data Science and AI Engineering

1.3 Tools for Machine Learning

Python has several modules that handle the different stages of the machine learning model development pipeline:

1. Data preprocessing: `pysql`, `pandas`
2. Exploratory data analysis: `pandas`, `numpy`, `matplotlib`
3. Optimisation: `scipy`
4. Implementation: `scikit-learn` (supervised and unsupervised methods), `keras`, `pytorch` (deep learing)

1.4 Scikit-learn

The basic syntax for using supervised learning models in `scikit-learn` follows a standard workflow:

1. **Split the data:**

```
x_train, X_test, y_train, y_test = train_test_split(X, y, test_size=...)
```

2. **Import and initialise the model:**

```
from sklearn import svm
model = svm.SVC(...)
```

3. **Train the model:**

```
model.fit(X_train, y_train)
```

4. **Make predictions:**

```
predictions = model.predict(X_test)
```

5. **Evaluate performance:** Use a confusion matrix to assess classification accuracy:

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, predictions)
```

6. **Optional – Save the model:** You can serialise the trained model using `pickle`:

```
import pickle
with open("model.pkl", "wb") as f:
    pickle.dump(model, f)
```

Whether this step is necessary depends on the context and industry practice.

2 Linear Regression

Regression is a type of supervised learning model. It models a relationship between a continuous target variable and explanatory features.

2.1 Introduction to Regression

Definition

Linear regression models the relationship between a response variable Y and one or more features $X_1, \ldots, X_p$. In the case of simple linear regression with one predictor X, the model assumes:

$$Y \approx \beta_0 + \beta_1 X$$

where β_0 is the intercept and β_1 is the slope.

Coefficient Estimation

Given training data $(x_1, y_1), \ldots, (x_n, y_n)$, the goal is to estimate the coefficients β_0, β_1 such that the fitted values are as close as possible to the observed values. This is done by minimizing a norm of the residual vector:

$$\beta_{\min} = \arg\min_{\beta \in \mathbb{R}^2} \|\mathbf{Y} - \mathbf{X}\beta\|_p^p$$

For ordinary least squares (OLS), we use the 2-norm ($p = 2$), leading to the minimisation of the residual sum of squares (RSS). The OLS solution yields closed-form expressions for $\hat{\beta}_0$ and $\hat{\beta}_1$. For least absolute deviations (LAD), we take $p = 1$. For Chebyshev or minimax regression, we take $p = \infty$, minimising the maximum residual.

Population vs Sample Regression

In the real world we almost always do not have a maximal data set. This means that we do not have data for every single item in the population, all we can hope to obtain are samples. The fitted regression line from a sample *estimates* the *population regression line*:

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

where ε is the irreducible error due to unobserved factors. Since we cannot observe the full population, we fit the model on a sample and obtain estimates $\hat{\beta}_0, \hat{\beta}_1$. However, since we only observe a sample, the estimates $\hat{\beta}$ vary from sample to sample[1]. The sample regression is thus an estimate of the population regression. If we had access to all population data, the OLS minimisation would yield the true β_0, β_1.

Sampling and the Central Limit Theorem

Across repeated samples of size n, we obtain different estimates of β, say $\hat{\beta}^{(1)}, \ldots, \hat{\beta}^{(n)}$. Their average converges to the true coefficient $\bar{\beta}$ as $n \to \infty$, due to the central limit theorem:

$$\hat{\beta} - \bar{\beta} \xrightarrow{d} \mathcal{N}(0, \sigma_*)$$

Standard Error and Inference

The variance of the coefficient estimates across samples is the *standard error*:

$$SE(\hat{\beta}_i) = f(\sigma^2, x_j, \bar{x})$$

where σ^2 is estimated from the data via:

$$\hat{\sigma}^2 = \frac{RSS}{n-2}$$

The standard error quantifies the variability of the estimated coefficient under repeated sampling. If we observe a high standard error then it means that we do not see replicability of the relationship as we vary our sample, which may suggest that the relationship between the features and target is ephemeral and noisy, so any forecast or inference made using it will be unreliable.

2.2 Multiple Linear Regression

Model Definition

Multiple linear regression generalises simple linear regression by modelling a response variable Y as a linear combination of multiple predictors $X_1, \ldots, X_p$:

$$\hat{Y} = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \cdots + \theta_p X_p = \mathbf{X}\boldsymbol{\theta}$$

where:

- $\mathbf{X}$ is the design matrix including a column of ones for the intercept,
- $\boldsymbol{\theta}$ is the parameter vector including θ_0 (intercept) and $\theta_1, \ldots, \theta_p$ (slopes).

For $p = 1$, the model defines a line as with the usual linear regression, for $p = 2$, the model defines a plane, for $p > 2$, the model defines a hyperplane in $\mathbb{R}^{p+1}$.

Estimation via Least Squares

As before the $\boldsymbol{\theta}$ coefficients are the population coefficients, but in reality we can only estimate these. These are the sample coefficients $\hat{\boldsymbol{\theta}}$ which are obtained by minimising the mean squared error (MSE), but there is no closed form solution to this minimisation problem and so numerical methods are used. For large datasets, iterative methods such as gradient descent can also be used to minimise MSE.

Prediction and Residuals

Predictions for a new data point $\mathbf{x}$ are given by:

$$\hat{y} = \mathbf{x}^\top \hat{\boldsymbol{\theta}}$$

The residual error for observation i is:

$$\varepsilon_i = y_i - \hat{y}_i$$

The squared residuals are used in minimising the MSE and finding the optimim sample coefficients.

[1] The goal of statistical inference is to understand how close $\hat{\beta}$ is to the true β and quantify this uncertainty.

Model Complexity and Overfitting

Including more variables increases model flexibility but can lead to overfitting, where the model captures noise rather than signal. A balance is needed:

- Remove redundant, highly correlated (collinear) variables.
- Choose variables that are interpretable, uncorrelated, and strongly related to Y.

Categorical Variables

Categorical predictors must be encoded numerically:

- Binary variables: encoded as 0/1.
- Multiclass variables: use one-hot encoding (create a dummy variable for each class).

What-If Analysis

The model can be used for counterfactual predictions by altering input features. However, what-if analysis may be invalid if:

- Scenarios lie far outside the training data (extrapolation),
- Variables are collinear — changing one realistically requires changing another.

2.3 Logistic Regression

Model Definition

Logistic regression models the probability that a binary outcome $Y \in \{0, 1\}$ occurs, given features $X_1, \ldots, X_p$. The model assumes that the log-odds (logit) of the probability is a linear function of the input:

$$\log\left(\frac{\mathbb{P}(Y = 1 \mid X)}{\mathbb{P}(Y = 0 \mid X)}\right) = \theta_0 + \theta_1 X_1 + \cdots + \theta_p X_p = \mathbf{X}^\top \boldsymbol{\theta}$$

Solving for the probability gives the sigmoid function:

$$\hat{p} = \mathbb{P}(Y = 1 \mid X) = \sigma(\mathbf{X}^\top \boldsymbol{\theta}) = \frac{1}{1 + e^{-\mathbf{X}^\top \boldsymbol{\theta}}}$$

This maps the linear combination of features to the interval $(0, 1)$, giving a probability.

Learning the Parameters

To fit the model onto data we use maximum likelihood estimation to learn the $\boldsymbol{\theta}$ parameters. The optimal parameters $\hat{\boldsymbol{\theta}}$ are found by solving:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$$

where $\mathcal{J}$ is negative log likelihood. This optimisation is typically performed using gradient descent or related numerical techniques.

Prediction and Decision Boundary

The model returns a probability $\hat{p}$. To make a classification, we choose a threshold $\tau \in (0, 1)$ (which is usually $\tau = 0.5$) and define:

$$\hat{Y} = \begin{cases} 1 & \text{if } \hat{p} \geq \tau \\ 0 & \text{otherwise} \end{cases}$$

This tells us that if $\mathbb{P}(Y = 1|X) < 0.5$ then $\hat{Y}$ is classified as belonging to class 0. The threshold can be fine tuned to match reality, since most phenomena are not cleanly 50:50. For example fraud may occur one time out of 99 so using a 0.5 threshold would clearly be unwise since it would class everything as not-fraud.

Interpretability

Each coefficient θ_j measures the change in the log-odds of $Y = 1$ for a unit increase in X_j, holding other variables fixed. Larger magnitudes indicate stronger influence. Importantly, this allows us to conduct inference involving continuous change on a discrete classification problem.

Use Cases

Logistic regression is used when:

- The target variable is binary.
- Probabilities, not just classifications, are required.
- Model interpretability is important.

2.4 Nonlinear and Polynomial Regression

Motivation

In many real-world datasets, the relationship between features X and the target Y is not well captured by a straight line. Nonlinear regression models are used when such relationships require curvature or more complex forms.

Nonlinear Regression: Definition

Nonlinear regression models the relationship between Y and the inputs $X_1, \ldots, X_p$ via a nonlinear function:

$$\hat{Y} = f(X_1, \ldots, X_p; \boldsymbol{\theta})$$

where f is nonlinear in its parameters $\boldsymbol{\theta}$. Common forms include:

- Exponential: $\hat{Y} = \theta_0 e^{\theta_1 X}$
- Logarithmic: $\hat{Y} = \theta_0 + \theta_1 \log(X)$
- Sinusoidal: $\hat{Y} = \theta_0 + \theta_1 \sin(\theta_2 X)$

Polynomial Regression

Polynomial regression models Y as an n-th degree polynomial in a single variable X:

$$\hat{Y} = \theta_0 + \theta_1 X + \theta_2 X^2 + \cdots + \theta_n X^n$$

This model is nonlinear in the input but linear in the parameters, so it can be recast as:

$$\hat{Y} = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \cdots + \theta_n X_n$$

where $X_k = X^k$, allowing the use of ordinary least squares.

Fitting and Optimisation

For models linear in parameters (e.g., polynomial), parameters $\boldsymbol{\theta}$ can be fitted using standard linear regression:

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$$

For truly nonlinear models, where f is nonlinear in $\boldsymbol{\theta}$, closed-form solutions do not exist. Instead, we minimise a loss (usually MSE) using iterative optimisation techniques, such as:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (y_i - f(x_i; \boldsymbol{\theta}))^2$$

This is solved numerically using algorithms like gradient descent.

Model Selection and Overfitting

Polynomial models of high degree can always perfectly fit any finite dataset (interpolate all points), but this leads to overfitting. Overfit models capture noise rather than signal.

Visual Model Selection

To identify whether a nonlinear model is needed:

- Plot scatter diagrams of Y vs. X_i
- Look for nonlinearity (e.g., curvature, saturation, periodicity)
- Choose candidate functions: polynomial, exponential, logarithmic, etc.

Examples of Nonlinear Forms

- Exponential growth: GDP, investment returns
- Logarithmic growth: Diminishing returns in productivity
- Sinusoidal: Seasonal patterns, temperature cycles

Beyond Parametric Forms

When no closed-form function is known, or when relationships are nonlinear and defy any known non-linear archetypes, meaning we cannot hope to represent the relationship via some standard paremetric, mathematical equation, we require more advanced methods such as

- Decision trees
- Random forests
- Support vector machines
- Neural networks
- k-nearest neighbours
- Gradient boosting

These are non-parametric regressors that adapt to complex patterns without requiring an explicit functional form.

3 Tree-based methods