1. Outliers: It is a data point that differs from the observations in the dataset. It maybe due to variability in data or error.

How to detect:

1. Statistical Methods:

1. **Z-Score**: If a data point’s z-score > 3 or < -3, it’s often considered an outlier.

Ex.

from scipy.stats import zscore

z\_scores = zscore(data)

outliers = np.where(np.abs(z\_scores) > 3)

1. **IQR (Interquartile Range)**:

Ex.

Q1 = data.quantile(0.25)

Q3 = data.quantile(0.75)

IQR = Q3 - Q1

outliers = (data < (Q1 - 1.5 \* IQR)) | (data > (Q3 + 1.5 \* IQR))

b.. Visualization:

* Boxplots
* Scatter plots
* Histogram

How to Handle/Remove:

1. **Remove them** if they are errors or noise.
2. **Cap or floor them** (Winsorizing).
3. **Transform** (e.g., log transform can reduce the effect).
4. **Replace** with median or mean (if justified).
5. **Keep them** if they're meaningful (e.g., rare diseases, fraud).

**Libraries used:**

1.Pandas

For basic data manipulation and statistical outlier handling (e.g., IQR).

2.Numpy

Used with statistical methods (like Z-score).

3.SciPy

For Z-score calculation, distributions.

4.Scikit-learn

Advanced outlier detection using ML models:

5.PyOD

Specialized library for outlier detection, with many algorithms.

6.Matplot/Seaborn/Plotly

For visualization: boxplots, scatterplots, etc.

2) Overfitting and Underfitting:

1. Overfitting: Model learns the training data **too well**, including noise. Fails on new/unseen data.

How to detect:

* Use **train/test/validation** splits or **cross-validation**
* Plot **learning curves** (error vs training set size)
* Monitor **validation accuracy/loss**

How to Avoid:

* + - * **Cross-Validation**
      * **Simpler Models** (e.g., fewer layers or trees)
      * **Regular:**

L1/L2 in linear models

Dropout in neural networks

* + - * **Pruning** (in decision trees)
      * **Early Stopping**
      * **More Data** (data augmentation)
      * **Ensemble Methods** (Random Forest, Bagging)

1. Underfitting: Model is **too simple** and fails to capture underlying patterns.

How to Avoid:

* **Use more complex models**
* **Feature engineering**
* **Reduce regularization**
* **Train longer**
* **Add interactions or polynomial features**

Libraries:

1. Scikit-learn:

Train/test split, models, regularization, pipelines.  
Regularization: Ridge, Lasso, etc.

Cross-validation: cross\_val\_score, GridSearchCV

Ex.

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.linear\_model import Ridge, Lasso

from sklearn.ensemble import RandomForestClassifier

1. Keras / TensorFlow:

Dropout, early stopping, batch normalization for neural networks.  
Ex.

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

from tensorflow.keras.callbacks import EarlyStopping

1. XGBoost / LightGBM / CatBoost:

Tree-based models with built-in regularization and early stopping.  
Ex.

import xgboost as xgb

from xgboost import XGBClassifier

1. MLflow / Optuna / Hyperopt:

For tuning and reducing overfitting via automated hyperparameter search.

## 

## **When to Use:**

* **Outlier Detection**: During **EDA (Exploratory Data Analysis)** and **data preprocessing**.
* **Overfitting/Underfitting Checks**: During **model training and validation** phases.