Figures for ML

This Document contains all figures used I used in ML

September 2024

1 Semester 1

1.1 Advanced Programming

]

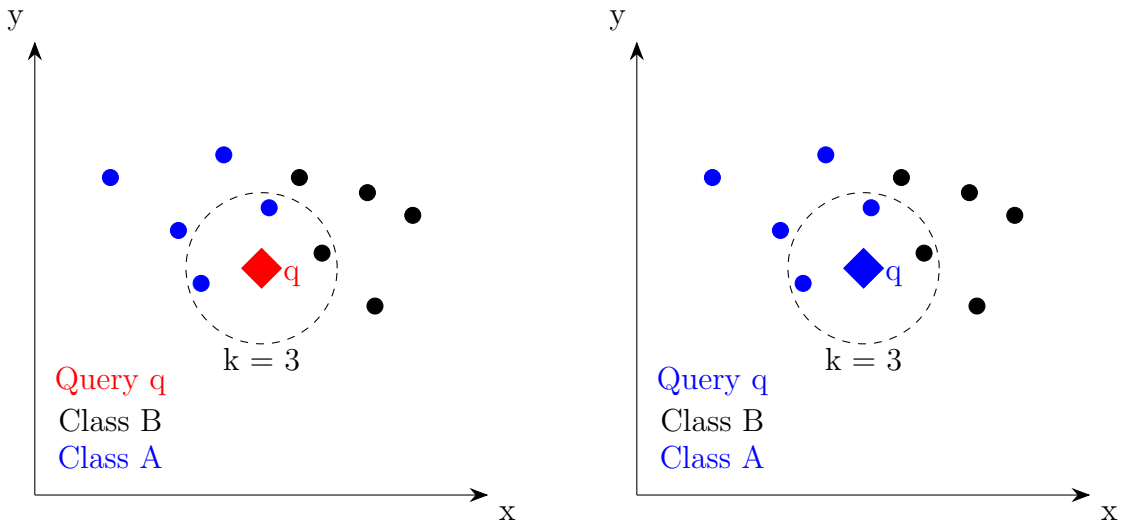

Figure 1: Query point q before and after performing kNN

1.1.1 kD-Trees

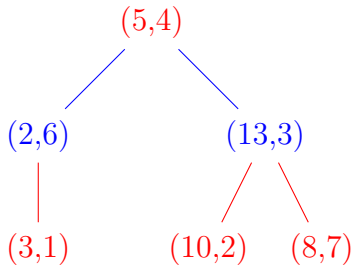

Figure 2: kD-tree from set $D = [(5,4),(2,6),(13,3),(8,7),(3,1),(10,2)]$ with $k = 2$ dimensions

1.1.2 kNN with kD-Trees

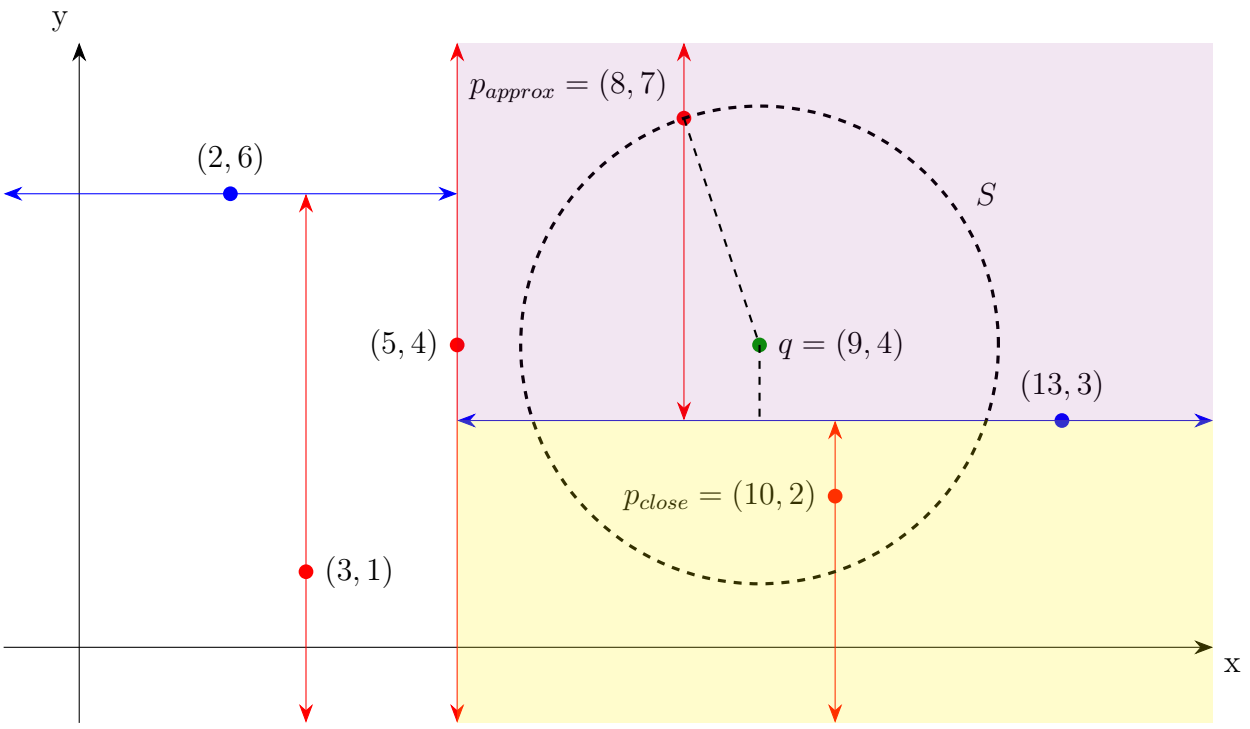

Figure 3: From point $(5, 4)D$ is split in half and an initial depth first search on the right branch of fig. 2 is performed and leads to p_{approx}. BWB test with sphere S shows that p_{close} is not located in the purple bin and backtracking is performed

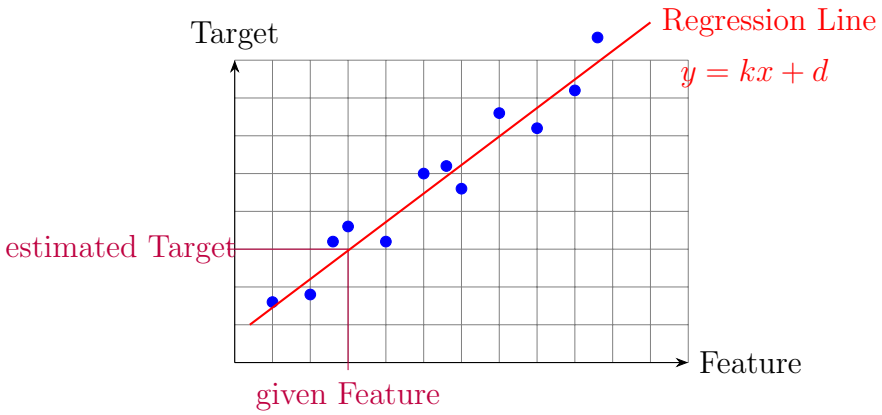

Figure 4: This figure illustrates a regression model applied to a dataset. The blue data points represent observations, such as the relationship between hours studied (Feature) and test scores (Target). The red regression line models this relationship, showing how test scores generally increase with more hours studied. The purple lines demonstrate an example where a specific number of study hours is the input. The expected test score can be estimated by projecting onto the regression line.

1.2 Machine Learning 1

1.2.1 Assignment 2

1.3 Regression

1.4 Classification

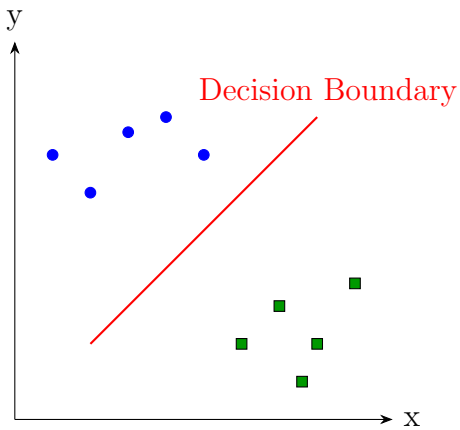

Figure 5: This figure depicts a classification problem involving two classes. Blue circles represent one class, such as emails labeled "Spam," and green squares represent another class, like "Not Spam." The red decision boundary separates the two classes based on features. The model uses this boundary to classify new emails as either Spam or Not Spam.

1.5 Features

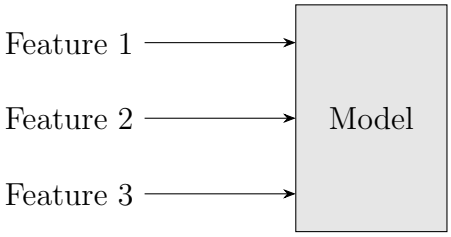

Figure 6: This diagram shows three features being input into a machine learning model. For example, in predicting house prices, Feature 1 could be the size of the house in square meters, Feature 2 the number of bedrooms, and Feature 3 the location. These features are fed into the model.

1.6 Targets

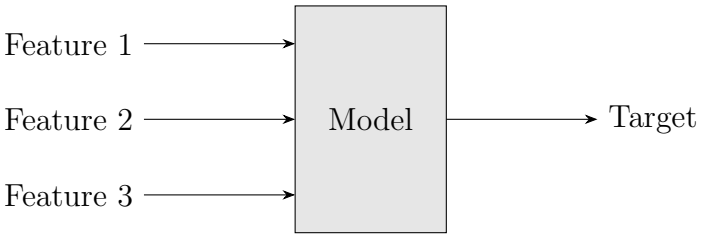

Figure 7: Extending figure 3, this figure illustrates the model producing a target output from the input features. After inputting the features, the model outputs a target value, such as the estimated price of the house in Euros.

1.7 Supervised Machine Learning Workflow

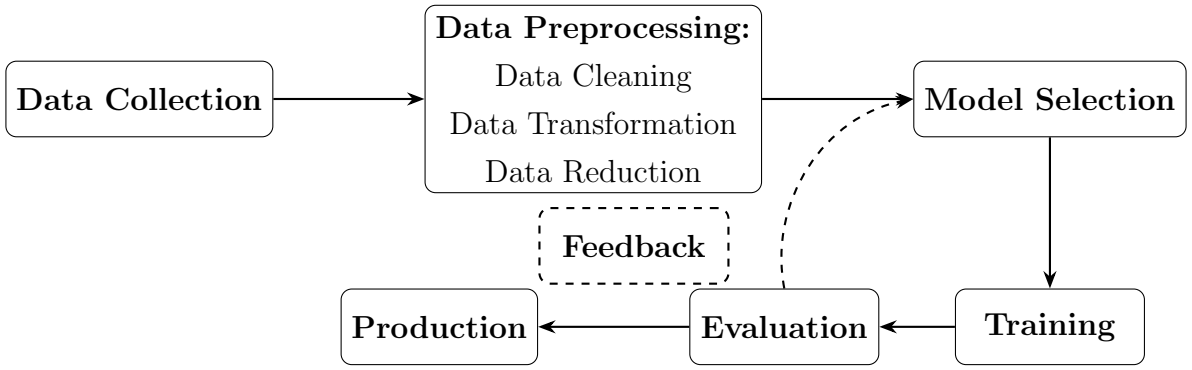

Figure 8: This flowchart represents the supervised machine learning workflow. The dashed feedback loop indicates that if the evaluation shows low accuracy, the model selection or training process may be revisited to improve performance.

The steps are:

1. **Data Collection**: Gather images labeled with the objects they contain (e.g., cats and dogs).
2. **Data Preprocessing**: Resize images and normalize pixel values.
3. **Model Selection**: Choose an appropriate model, like a convolutional neural network (CNN).
4. **Training**: Train the CNN using the labeled images.
5. **Evaluation**: Assess the model's accuracy in classifying new images.
6. **Prediction**: Use the trained model to classify unlabeled images as either cats or dogs.

1.7.1 Assignment 3

1.8 Train-Test Split

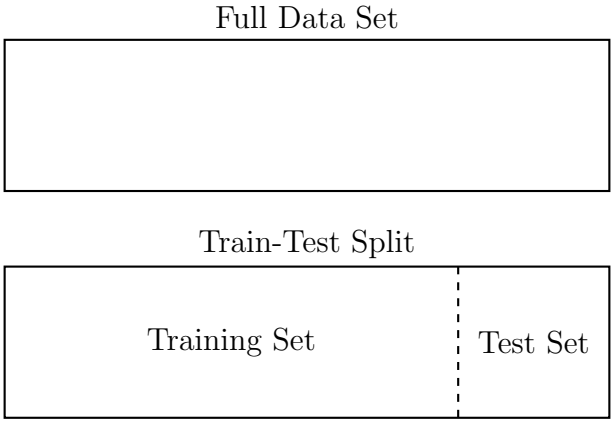

Figure 9: The Train-Test split is applied on data to split the test in training data and test data that is unseen to the model to validate it. A typical split ratio is 80% training data and 20% test data.

It is important to shuffle the data before processing because of many reasons. Some of the most imortant are:

- **Break order based patterns:** shuffling prevents models from learning unintended dependencies on data order (e.g. is often gathered in certain groups).
- **Reduce bias and overfitting:** ensure diverse data represantation in every batch of training data.
- **Generalization and effectiveness:** to ensure the model is able to learn meaningful patterns and perform well on unseen data .

1.9 Mean Absolute Error

The formula for Mean Absolute Error (MAE) is given by:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Where:

- n is the total number of observations.
- y_i is the actual value for the i-th observation.
- $\hat{y}_i$ is the predicted value for the i-th observation.
- $|y_i - \hat{y}_i|$ is the absolute difference between the actual and predicted values.

MAE measures the average magnitude of errors between the predicted values and actual values, treating all errors equally. For example if a model wants to predict the location of an object in an image. The actual coordinates of the top-left corner are $(50, 100)$, while the predicted coordinates are $(55, 90)$.

To calculate the Mean Absolute Error (MAE):

- **Actual Coordinates**: $(50, 100)$
- **Predicted Coordinates**: $(55, 90)$
- Absolute error for the x-coordinate: $|50 - 55| = 5$
- Absolute error for the y-coordinate: $|100 - 90| = 10$

The MAE for this prediction is:

$$MAE = \frac{5 + 10}{2} = 7.5$$

This means the average error in object location prediction is 7.5 pixels.
While MAE treats all errors equally by calculating the average of absolute differences and therefore makes it less sensitive to outliers, the Root Mean Squared Error (RMSE) squares the errors before averaging, giving more weight to larger errors. This means RMSE is more sensitive to outliers, as large deviations are amplified.

1.10 Accuracy

$$Accuracy = \frac{Correct\,Predictions}{All\,Predictions} = \frac{TP + TN}{TP + TN + FP + FN}$$

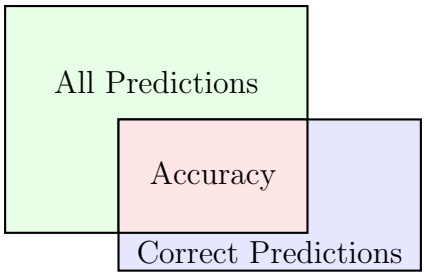

Figure 10: This diagram shows the overlap between all predictions (green) and correct predictions (blue), with the overlapping area corresponding to the accuracy. The bigger the overlap is, the higher the model's accuracy. Very sensitive to class imbalance (more positives than negatives).

1.11 Confusion Matrix (Snippet)

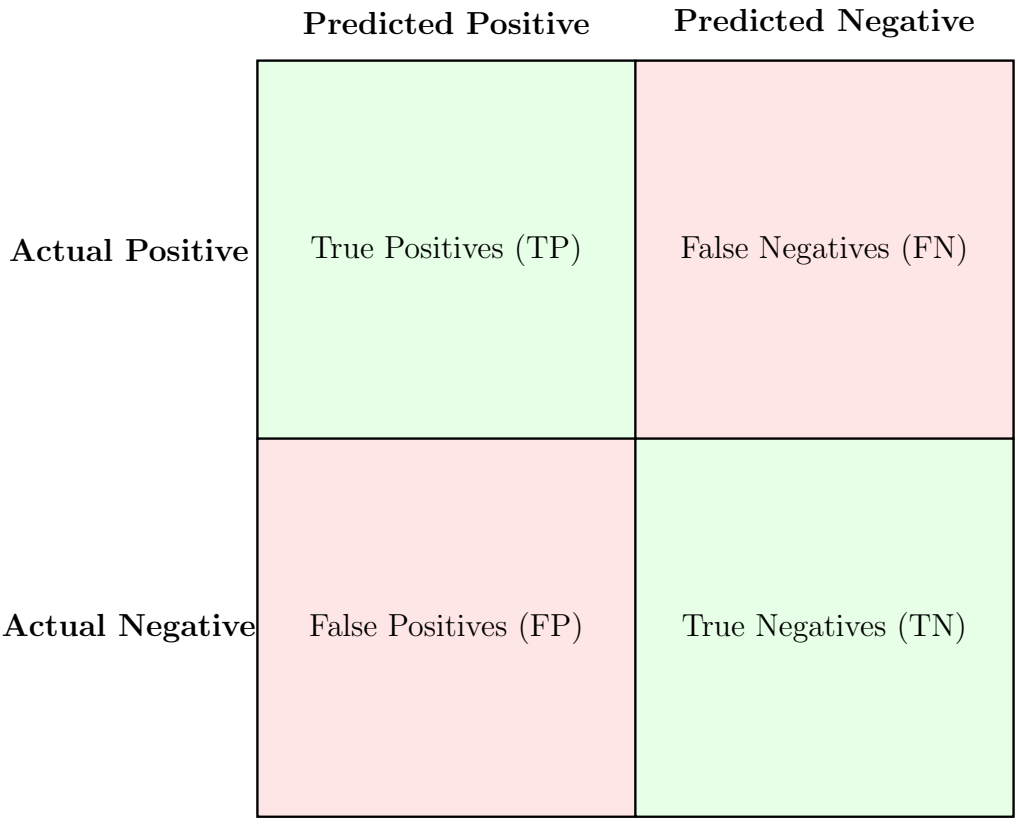

Figure 11: Confusion Matrix is a table to evaluate the performance of a model by comparing predicted labels with actual labels. The rows are actual and the columns are predicted labels

The steps are:

1. **True Positives**: Correctly predicted positive cases (e.g., cars that are labeled as cars).
2. **False Positives**: Incorrectly predicted positive cases (e.g., buses predicted as cars).
3. **True Negatives**: Correctly predicted negative cases (e.g., buses predicted as not cars).
4. **False Negatives**: Incorrectly predicted negative cases (e.g., cars predicted as not cars).

1.11.1 True Positive Rate (Sensitivity)

The True-Positive Rate (TPR) measures the correctly identified labels. TPR is crucial when missing positive cases is costly (e.g. stop sign detection for self driving car). Not sensitive to True Negatives. It is defined by:

$$TPR = \frac{TP}{TP + FN}$$

1.11.2 True Negative Rate (Specicivity)

The True-Negative Rate measures the proportion of actual negatives correctly identified. A high TNR is crucial when avoiding false positives is crucial (e.g. marking credit card transactions as non fraud). It is defined by:

$$TNR = \frac{TN}{TN + FP}$$

1.11.3 Assignment 4

Explain “hyper parameters“ and “K nearest neighbours“ in your own words.

1.12 Hyperparameters

Hyper parameters are external configuration variables to manage machine learning training. They are determined through tests and are not obtained from the actual data. They determine how a model learns and are therefore critical for fine-tuning. Examples include:

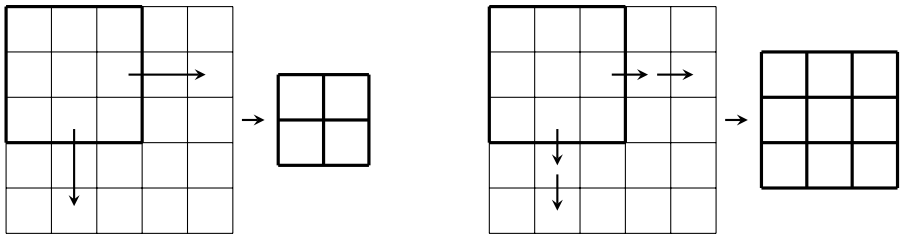

Figure 12: Example of stride

- Stride: Step size of filter
- Filter Size: Size of sliding window
- Learning rate: Step size towards minimum of loss function in one iteration
- Batch size: Number of training samples used in one iteration

1.13 K-Nearest-Neighbours

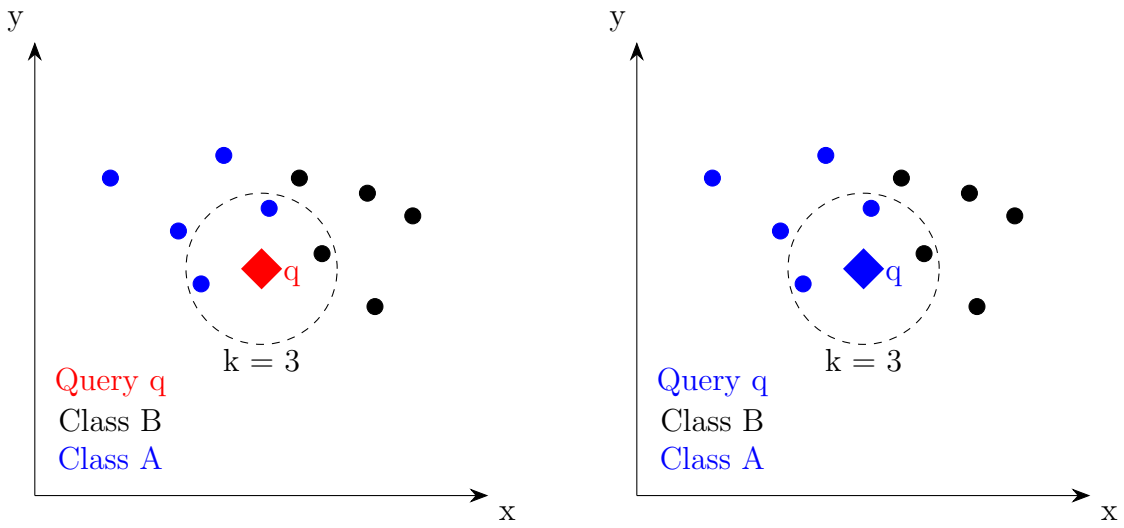

Figure 13: Query point q before and after performing kNN

K-nearest-neighbours (KNN) is a machine learning algorithm. it classifies points based on majority vote of it's k nearest neighbours. K and the distance metric are hyper-parameters the output depends heavily on it. KNN works works by calculating the distance from each point in the dataset to the query point (lazy learning).

- k: The value of k significantly affects performance. A small K is sensitive to noise, while a large K might over-generalize.
- Distance Metric: is used to calculate how close two points are (e.g Manhattan, Euclidian)

1.13.1 Assignment 5

1.14 Missing Data

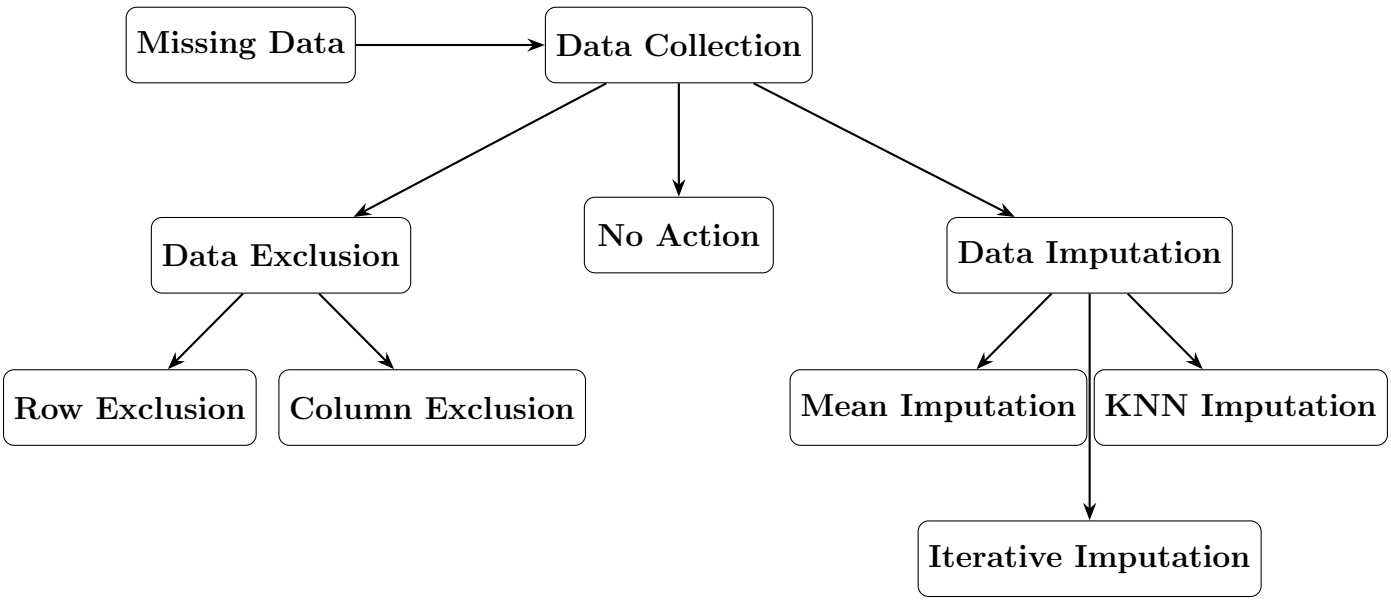

Figure 14: Methods for Handling Missing Data

When dealing with missing data there are several strategies for different scenarios. They can broadly be categorized into collection, exclusion, imputation:

- **Data Collection**: best way to adress missing data is to collect it through further data gathering
- **Data Exclusion**: if collection more data is not feasable either **rows** or **cols** can be dropped. But this can introduce bias.
- **Data Imputation**: missing values can be filled by:
  - **Mean Imputation**: fill with mean value of the column.
  - **Iterative Imputation**: use a regression model to to predict the missing values.
  - **KNN Imputation**: use knn algorithm to predict missing values based on similar data points.
- **No Action**: in some case algorithms can handle missing values.