

**ENGR 5005: Machine Learning for Engineering Applications**

Final Coding Project

Muhammad Eesa Khan | 100742997

March 24.2025

Contents

[I. Abstract 4](#_Toc193744799)

[II. Background Theory 5](#_Toc193744800)

[i. Linear & Logistic Regression 5](#_Toc193744801)

[ii. Decision Tree 6](#_Toc193744802)

[iii. K-Nearest Neighbours 7](#_Toc193744803)

[iv. Bayesian Classifier 7](#_Toc193744804)

[III. Breast Cancer Data 8](#_Toc193744805)

[A. Dataset Background 8](#_Toc193744806)

[B. Linear & Logistic Regressions 8](#_Toc193744807)

[C. Decision Trees 9](#_Toc193744808)

[D. KNN 9](#_Toc193744809)

[E. Bayesian Classifier 9](#_Toc193744810)

[F. Discussion & Accuracy Comparison 10](#_Toc193744811)

[IV. Mushroom Data 10](#_Toc193744812)

[G. Dataset Background 10](#_Toc193744813)

[H. Linear & Logistic Regressions 11](#_Toc193744814)

[I. Decision Tree 11](#_Toc193744815)

[J. KNN 11](#_Toc193744816)

[K. Bayesian Classifier 12](#_Toc193744817)

[L. Discussion & Accuracy Comparison 12](#_Toc193744818)

[V. Robotic Data 13](#_Toc193744819)

[M. Dataset Background 13](#_Toc193744820)

[N. Linear & Logistic Regressions 13](#_Toc193744821)

[O. Decision Trees 14](#_Toc193744822)

[P. K-Nearest Neighbors 14](#_Toc193744823)

[Q. Bayesian Classifier 14](#_Toc193744824)

[R. Discussion & Accuracy Comparison 15](#_Toc193744825)

[VI. Personal Data Selection # 1 – Obesity Levels 16](#_Toc193744826)

[S. Dataset Background 16](#_Toc193744827)

[T. Linear & Logistic Regressions 16](#_Toc193744828)

[U. Decision Trees 17](#_Toc193744829)

[V. KNN 17](#_Toc193744830)

[W. Bayesian Classifier 17](#_Toc193744831)

[X. Discussion & Accuracy Comparison 18](#_Toc193744832)

[VII. Personal Data Selection #2 - Advertising 18](#_Toc193744833)

[Y. Dataset Background 18](#_Toc193744834)

[Z. Linear & Logistic Regressions 19](#_Toc193744835)

[AA. Decision Trees 19](#_Toc193744836)

[BB. KNN 19](#_Toc193744837)

[CC. Bayesian Classifier 20](#_Toc193744838)

[DD. Discussion & Accuracy Comparison 20](#_Toc193744839)

[VIII. Conclusion & Future Implementation 21](#_Toc193744840)

[Other 22](#_Toc193744841)

# Abstract

This project consists of systematic derivation and inspection of fundamental machine learning models for the classification and regression tasks using a furniture-based attribute data set. The primary objective was to implement fundamental machine learning algorithms by hand—without employing libraries such as Scikit-Learn—so that a clear understanding of their mathematical and computational underpinnings is gained. The models developed are Bayesian Classifiers, K-Nearest Neighbors (KNN), Logistic Regression, and Linear Regression, all well-coded and tested for performance.

A rigorous data preprocessing pipeline was imposed, such as categorical encoding, feature normalization, and missing value imputation to enhance the accuracy and generalizability of models. The Bayesian Classifier was implemented utilizing probabilistic estimation techniques, while the KNN model employed Euclidean distance-based nearest neighbors. Both the Logistic and Linear Regression models employed gradient descent-based optimization. All models were validated utilizing proper performance measures: classification accuracy in discrete prediction cases and mean squared error (MSE) in regression scenarios.

This work emphasizes the worth of machine learning model design from first principles, favoring deep, conceptual understanding of algorithmic process over more conventional black-box deployments. The techniques and outcomes presented here are a foundation for future applications in data-driven decision-making and predictive analytics.

# Background Theory

The section presents the background for each of the classification model and its importance on implementing the usage of any machine learning library. The report will outline the following methodologies:

* Linear & Logistic Regression
* Decision Tree
* K-Nearest Neighbors
* Bayesian Classifier

# Linear & Logistic Regression

**Linear regression,** a very common and relatively simple model, is utilized by many programmers as a supervised learning algorithm to predict continuous values. It has a linear relationship, modeled by:

A black letter on a white background

AI-generated content may be incorrect.

The equation of a straight line will create a linear relationship between the input and the output variables – the primary goal is to minimize the difference between the actual values and the predicted values. In the equation listed above,

* y – predicted, final value
* x – input value
* m – the slope or weight
* b - y-intercept

To train the model, the programmers will adjust the weight (m) to reduce the error between the predicted and the actual values. To measure the displacement and deviation between the real and predicted values, a **mean squared error** is preformed. It is modeled by:

A black and white symbol

AI-generated content may be incorrect.

Where:

* Ypred – predicted values
* Ytrue – actual value
* m –number of samples

The model essentially accepts a new input, utilizes the weight of the equation to create a prediction within that linear line. This technique is most effective during 1to1 relationships between output and input values.

**Logistic Regression** is also a commonly used model, which is utilized for binary classification problems – where the target values are fixed from a set. The model predicts probabilities and assigns the target value to an item from a list. The regression uses the sigmoid function to create outputs between 0 and 1 which can be deceived as probabilities.

A math equation with numbers

AI-generated content may be incorrect.

Where:

-x – the coefficient that determines the corresponding input feature contributes to the models prediction.

The equation represents the probability of an instance going over a turning point to decipher between 2 outputs i.e. a 0.5 threshold indicates that values over represent an output of 1 while lower represents the 2nd output.

# Decision Tree

A decision tree is a supervised learning algorithm which is utilized for both classification and regression tasks – similar to a tree diagram, the decision tree consists of:

* Nodes – represent a decision
* Branch – the outcome of a decision(node)
* Leaf – the final label or final value deciphered.

The decision trees will split the data based on the criteria that can split the data and provide the most information. The two methodologies for the data involve **Gini Impurity** & **Entropy**; Gini Impurity measures the likelihood of an incorrect classification while the entropy measures the randomness/uncertainty in the data set. The features will split the data continuously until a final verdict or a stopping condition is met – this is called the maximum death where the split can not progress further. To prevent phenomena such as overfit, the programmers can pre or post pruning which removes branches to simplify the model.

# K-Nearest Neighbours

The K-Nearest Neighbors (KNN) algorithm is a simple but powerful supervised learning method used for classification and regression tasks. It is an instance-based and non-parametric learning algorithm since it does not make any assumptions about the data distribution and stores the training set explicitly. In contrast to learning explicit parameters or functions, KNN uses the similarity between instances to predict, and therefore it is very useful for pattern recognition and anomaly detection.

KNN works by finding the K nearest data points (nearest neighbors) to a given target input sample and using them to calculate the output. In classification, the algorithm gives the most frequent class label of the K neighbors to the input sample as the output. In regression, the output is usually the mean (or weighted mean) of the values of the K nearest neighbors. The choice of K (number of neighbors) is critical since too small a K renders the model overly sensitive to noise, while too large a K could result in prediction that is too smoothed.

In the estimation of "closeness" between data points, KNN utilizes distance measures; below is the Euclidean Distance (most common method):

A red square with numbers and symbols

AI-generated content may be incorrect.

# Bayesian Classifier

The Bayesian Classifier is a Bayes' Theorem probabilistic model based on prior knowledge and observed evidence that provides a mathematical prediction framework. It is typically used in classification problems, particularly text classification (e.g., spam filtering, sentiment analysis) and medical diagnosis. The model is simple yet robust, with the capability to handle large datasets effectively and provide probabilistic predictions about class membership.

At the heart of Bayesian classification lies Bayes' Theorem, which states:

A math equation with black text

AI-generated content may be incorrect.

Where:

* P(Y|X) – the posterior probability (Y given input of X)
* P(X|Y) – the probability of observing X given class Y
* P(Y) – prior probability of class Y
* P(X) – Probability of input X

The main classifier utilized are the **Naïve Bayes Classifier**, **Gaussian Naïve Bayes** these are for features for independent given the class label.

# Breast Cancer Data

## Dataset Background

The Breast Cancer dataset contains 569 instances and 30 real-valued features, thus a multivariate dataset utilized primarily for classification in the health and medicine domain. The features are derived from digitized images of fine needle aspirates (FNA) of breast masses and refer to various properties of the cell nuclei in the images. Such properties help to discriminate between benign and malignant tumors.

## Linear & Logistic Regressions

In this section, I created both Linear & Logistic Regression models to analyze the Breast Cancer dataset. Utilizing Pandas, I was able to load the dataset, and rename the columns; additionally, I encoded the labels as binary values as this dataset is a classifier dataset. The features were normalized to allow all values at a standardized scale.

With an 80-20 train-test split, I shuffled the values to ensure a random set of data. The linear regression model optimized the weight parameters to allow for a minimal MSE (Mean Squared Error). The logistic regression model, designed for classification problems, utilized a gradient descent to optimize weights. The model achieved a very high accuracy, and I have strong confidence in the model distinguishing between malignant and benign cases. In this situation, the logistic regression is better suited for classification task.

## Decision Trees

Decision Tree classification was applied to the breast cancer data to classify tumor samples as malignant or benign. The model was trained to a depth of 5 to prevent overfitting while retaining the important patterns within the data. After the model was fit to the training set, it was tested against the test set and classified with an about 92.98% classification accuracy. The ability of the Decision Tree to create strong decision boundaries by feature splits helped it perform adequately in distinguishing the two classes. Despite the model's simplicity, the accuracy confirmed its robustness in executing this classification task.

## KNN

K-Nearest Neighbors (KNN) classifier was created from scratch for classifying the Breast Cancer dataset. The algorithm was trained to make a prediction of the class of a point by finding the k nearest points in the training set based on the Euclidean distance

The classifier achieved a high accuracy ratio of over 90%, which shows the performance of the classifier on this dataset. The high performance can be attributed to the well-separated features in the dataset, which allows the majority voting principle to result in accurate classification. The value of k=5 helped to reduce noise sensitivity without compromising bias-variance trade-off. However, KNN incurs growing computational cost with data size since every prediction includes computation of distances.

## Bayesian Classifier

The Bayesian Classifier involved the development of a model to forecast the Breast Cancer dataset – it was trained under the premise that the features are Gaussian distributed. To prep the dataset, the data was split into training and test sets and then train the model by approximating the probability of every class and its deviance of every feature relative to the mean. Predictions were generated by combining the probabilities with the likelihood to compute posterior probabilities, to finally assign each instance to a case.

The Gaussian Naïve Bayes produced a high accuracy which proves to be effective for this dataset. The Naïve Bayes assumes features to be independent – it classifies data without demanding extensive parameter optimization.

## Discussion & Accuracy Comparison

All of the five models—Linear Regression, Logistic Regression, Gaussian Naive Bayes, K-Nearest Neighbors (KNN), and Decision Trees—were very accurate in the Breast Cancer data. The most accurate of them were Logistic Regression and Gaussian Naive Bayes, both having more than 90% accuracy. Logistic Regression works exceptionally well with binary classification problems, effectively extracting class probabilities. Gaussian Naive Bayes also performed similarly, from its high assumption of feature independence as well as Gaussian distribution. While Decision Trees also performed well, the explicit interest of Logistic Regression in binary classification made it the highest-performing model here.

A close up of a number

AI-generated content may be incorrect.

# Mushroom Data

## Dataset Background

The dataset contains descriptions of 23 species of gilled mushrooms from the Agaricus and Lepiota families, based on physical characteristics. It is used for classification, where the task is to determine whether a mushroom is edible or poisonous. The dataset includes 8,124 instances and 22 features, with categorical data types. Some mushroom species are classified as definitely edible, while others are poisonous or of unknown edibility (which has been combined with the poisonous category). The dataset also contains missing values.

## Linear & Logistic Regressions

This code preprocesses a mushroom dataset to mark mushrooms as poisonous (1) or edible (0). The dataset contains categorical features, which are numerically encoded first by assigning each category a distinct integer value. The features are also normalized to the range of 0 to 1 to prevent issues such as division by zero during calculations. The dataset is then split into training and test sets, with an 80-20 split following preprocessing.

Two models, Linear Regression and Logistic Regression, are trained on the data. Logistic Regression is trained as a classification model, which produces a binary prediction (0 or 1) based on the learned parameters. The model is evaluated using accuracy, which is the proportion of correct predictions on the test set. Linear Regression is also applied for classification, though typically designed for regression tasks. It is forecasting continuous values in this case, clipped to -1 to 1 range to prevent gigantic errors at the time of mean squared error (MSE) calculation. Performance of both models is compared: Logistic Regression on accuracy and Linear Regression on MSE.

## Decision Tree

The Decision Tree works by recursively splitting the data based on the feature that provides the best Gini index, which is a measure of impurity. The tree continues to split until it either reaches the maximum depth specified or the labels in a node are homogeneous. The \_best\_split method evaluates all possible splits and selects the one with the lowest Gini index. The classifier then makes predictions by traversing the tree, following the splits until reaching a leaf node, where the majority class is assigned as the predicted label. The model's accuracy is computed by comparing the predicted labels to the true labels in the test set.

## KNN

The KNN classifier function works by calculating the Euclidean distance from a test sample to all the training samples. For one test sample, it identifies the `k` nearest neighbors such that `k=5` here, and then selects the most common class from the neighbors. The class which is most common among the neighbors is assumed to be the predicted label. The classifier is evaluated by calculating the accuracy, i.e., the proportion of accurate predictions in the test set. The accuracy score calculated is a gauge of how accurately the classifier performs on the test set.

## Bayesian Classifier

The Naive Bayes classifier is trained by first calculating the prior probability of each class (poisonous or not poisonous) and the probability of each feature given the class. These are stored in memory for use later. For prediction, the classifier calculates the log-sum of the class priors and the feature probabilities for each sample, choosing the most likely class. The model is evaluated according to accuracy, which is calculated as the number of correct predictions in the test set divided by the total. The accuracy score attained is indicative of how well the classifier is classifying the test data.

## Discussion & Accuracy Comparison

A white background with black text

AI-generated content may be incorrect.

The models were tried on the mushroom dataset and yielded varying performances. Linear Regression was 3.28% for mean squared error, which was not ideal for classification purposes because it is not particularly designed for discrete output and might struggle to deal with classification problems. On the other hand, Logistic Regression was 93.78% accurate, performing wonderfully with its probabilistic approach, particularly with binary classification, as it has the ability to capture the correlation between the input features and probability of belonging in a specific class. The Decision Tree classifier did wonderfully with the perfect accuracy at 100.00%, being able to capture non-linear decision boundaries and handle complex interactions among features. Its tree-like structure allows for natural splits distinguishing classes effectively. K-Nearest Neighbors (KNN) also performed extremely well at 99.94% classification accuracy, with good performance despite its computational complexity because it makes its decision based on data points' proximity and is sensitive to local trends. Naive Bayes scored a lower accuracy of 88.12%, which although still effective, signifies the weakness of the model in cases where feature independence assumptions are broken, especially with categorical data like the mushroom dataset. Decision Tree and KNN models tended to be more accurate overall because they can accommodate flexible and localized decision-making processes, while Logistic Regression performed well as a fast and simple probabilistic model.

# Robotic Data

## Dataset Background

This data set consists of force and torque readings from a robot at failure detection, and for each failure there are 15 samples of force/torque measurements collected within the same duration over a window of 315 ms. The data set holds 90 features capturing forces (Fx, Fy, Fz) and torques (Tx, Ty, Tz) at each measurement.

There are five classification problems in this data set related to different failure cases in the robotic operation

1. LP1 Grasp approach failures.

2. LP2: Part transfer failures.

3. LP3: Part position after transfer failure.

4. LP4: Ungrasp approach failures.

5. LP5: Motion failures with a part.

## Linear & Logistic Regressions

The data is split into features and labels, and factorization is performed on the encoding of the labels. The data set is then manually split between training and testing sets by invoking a custom function, `train\_test\_split\_manual`, to properly shuffle and partition the data.

Two machine learning algorithms, Linear Regression and Logistic Regression, are trained and compared. The Logistic Regression algorithm is trained to predict the failure labels, using a sigmoid activation function to give output and probabilities. The accuracy of the model on the test set is calculated. On the other hand, Linear Regression is used to predict the failure labels as continuous values and Mean Squared Error (MSE) between the predicted and actual values is used to evaluate the performance of the model. The comparison between these two models would make it clear whether classification or regression suits the dataset more.

## Decision Trees

The tree is grown by calculating the entropy of the target variable and selecting the feature and threshold that result in maximum information gain at every node. The tree recursively splits until a maximum depth is reached or all samples in a node have the same class.

After training, the tree is used to predict on the test set, and accuracy is assessed by comparing predicted and true labels. The Decision Tree classifier proves to be good at classifying robot failures from measured forces and torques, with the ability to handle hierarchical data and produce interpretable decisions making it highly suitable for the task.

## K-Nearest Neighbors

The classifier is trained by storing the training data, and in prediction, it calculates the Euclidean distance from each test example to all the training examples. Following training, the test set is predicted, and accuracy is calculated by comparing predicted labels with actual labels. The performance of the KNN classifier is then displayed, showing its ability in classifying robotic failures based on force and torque data. The simplicity of the KNN model and its ability to handle high-dimensional data make it a suitable model for this classification task.

## Bayesian Classifier

A Naive Bayes classifier is then trained by calculating class priors, means, and variances for every one of the classes. To avoid issues with zero probabilities, smoothing is included in the variances. In prediction, the likelihood of each test sample belonging to each class is calculated based on the Gaussian distribution. The class in which each sample has the highest posterior probability is selected. Predictions are compared against actual labels, and model accuracy is obtained.

The Naive Bayes classifier is evaluated and found to perform well for classification of robotic failure conditions based on force and torque measurements. With its simplicity, efficiency, and ability to function with high-dimensional data, the classifier is effective for this task. Despite making the independence assumption, it happens to perform decently in reality, especially on datasets where the features are decently conditionally independent.

## Discussion & Accuracy Comparison

|  |  |  |
| --- | --- | --- |
| **Data Set** | **Model** | **Accuracy (Decimal)** |
| LP1.data | Linear Regression MSE | 3.36 |
|  | Logistic Regression Accuracy | 0.227 |
|  | Decision Tree | 0.696 |
|  | KNN Classifier Accuracy | 0.787 |
|  | Naïve Bayes Classifier | 0.55 |
|  |  |  |
| LP2.data | Linear Regression MSE | 3.38 |
|  | Logistic Regression Accuracy | 0.15 |
|  | Decision Tree | 0.56 |
|  | KNN Classifier Accuracy | 0.6 |
|  | Naïve Bayes Classifier | 0.55 |
|  |  |  |
| LP3.data | Linear Regression MSE | 1.36 |
|  | Logistic Regression Accuracy | 0.29 |
|  | Decision Tree | 0.66 |
|  | KNN Classifier Accuracy | 0.64 |
|  | Naïve Bayes Classifier | 0.57 |
|  |  |  |
| LP4.data | Linear Regression MSE | 1.364 |
|  | Logistic Regression Accuracy | 0.47 |
|  | Decision Tree | 0.826 |
|  | KNN Classifier Accuracy | 0.860 |
|  | Naïve Bayes Classifier | 0.635 |
|  |  |  |
| LP5.data | Linear Regression MSE | 4.76 |
|  | Logistic Regression Accuracy | 0.58 |
|  | Decision Tree | 0.64 |
|  | KNN Classifier Accuracy | 0.66 |
|  | Naïve Bayes Classifier | 0.449 |

In the case of LP1 dataset, the maximum recorded accuracy was 0.787 using the KNN classifier, far above all the rest, including the Logistic Regression model whose accuracy was merely 0.227. Decision Tree model came in fairly high at 0.696, indicating that it can classify pretty well for this task. Naïve Bayes and Linear Regression came pretty low, the latter with an MSE of 3.36.

In the LP2 dataset, all the models were worse and Logistic Regression was worst at 0.15 accuracy. Naïve Bayes and KNN classifier were best with 0.6 accuracy, suggesting the possibility that the dataset might be hard for linear models to model.

For LP3, Linear Regression MSE was 1.36, which was an increase from previous datasets but still with the mixed results for classification models. KNN had the highest accuracy at 0.64, which was slightly better than Decision Tree at 0.66, although Naïve Bayes once again recorded a relatively lower result of 0.57.

For the LP4 dataset, the KNN classifier again performed the best with a performance of 0.860 followed closely by the Decision Tree at 0.826. Logistic Regression recorded an improvement by 0.47 accuracy from the previous datasets. Naïve Bayes also performed reasonably at 0.635 but still lagged behind the top two models.

Finally, the LP5 dataset achieved the worst performance overall. Linear Regression had the greatest MSE of 4.76, while Logistic Regression reached an accuracy of 0.58. The KNN classifier performed reasonably at 0.66, while Naïve Bayes performed badly at an accuracy of 0.449.

# Personal Data Selection # 1 – Obesity Levels

## Dataset Background

The obesity dataset is focused on predicting the degree of obesity in individuals from Mexico, Peru, and Colombia based on their diet and physical condition. Obesity is a health problem of the developing world with serious physical and psychological consequences, and it is therefore very important to investigate its determining factors and develop predictive models. The dataset contains 2111 records and 17 features including demographic, dietary, lifestyle, and physical activity variables. The target variable, "NObeyesdad," classifies people into different levels of obesity. The dataset is extremely beneficial for predicting obesity and preventing policies.

## Linear & Logistic Regressions

The obesity dataset was preprocessed through manual encoding of categorical variables and feature value normalization to enhance numerical stability. The dataset was subsequently split into a training set and a test set using a manual splitting function with an 80-20 distribution.

A Logistic Regression model was learned from scratch to make predictions of the level of obesity. The model learned its weights using gradient descent over 1,000 iterations with a learning rate of 0.01. The model did not do well with an accuracy of only around 18.5%. This suggests that obesity classification is not best handled by a linear decision boundary. The computed MSE was close to 1.97, and it shows considerable disparity between predicted and target values. This again proves that employing a linear solution is not very effective in considering the complexity of obesity classification because the target variable in this scenario happens to be categorical and not continuous.

## Decision Trees

The use of a bespoke Decision Tree classifier to label data from the `obesity.csv` dataset began with loading and preprocessing the data, with categorical features manually encoded as numerical labels. The feature set (`X`) and target variable (`y`) were then defined, and the features were normalized for improved numerical stability. A bespoke train-test split function was created, dividing the data into 80% training and 20% testing. Decision Tree algorithm was applied, such as the entropy calculation functions, determining the best feature to split on, and recursively building the tree by information gain. In training, tree building was performed by searching for the best splits to a depth of 5..

## KNN

A K-Nearest Neighbors (KNN) classifier was implemented from scratch, with k=5k=5, meaning the classification was determined by the majority class among the five nearest neighbors. The model computed the Euclidean distance between test samples and training data points to identify the closest neighbors. Predictions were made based on the most frequently occurring label among these neighbors. The model achieved an accuracy of 88%, indicating a significantly better performance compared to Naïve Bayes (61%). This improvement suggests that KNN is more suitable for this dataset, likely due to its ability to capture non-linear relationships and dependencies between features.

## Bayesian Classifier

In this implementation, the obesity dataset was preprocessed such that the categorical variables were encoded manually with dictionary mapping. This helped transfer categorical data to numerical values so that the data was ready to be analyzed. The dataset was then split between features (X) and the target variable (y), along with manual separation of data to train and test. The splitting function shuffled the dataset indices randomly using 80% of the data for training and 20% for testing to make unbiased evaluations of model performance.

The model produced class prior, mean, and variance predictions for every feature within each class. When it made predictions, it applied the Gaussian probability density function (PDF) and Bayes' theorem to compute the probability of each class and marked the class with the highest posterior probability.

## Discussion & Accuracy Comparison

The use of three different machine learning algorithms—K-Nearest Neighbors (KNN), Logistic Regression, and Decision Tree—on the dataset returned varying performance measures. Both Logistic Regression and KNN had accuracy of 18.48%, indicating that the two models did not properly recognize the underlying structures in the data. The linearity and simplicity of KNN, which relies on nearest neighbors, and of Logistic Regression can have been constraining factors because the two models would not have been in a position to handle the complexity and non-linearity in the data effectively. The Decision Tree model, however, performed better with an accuracy of 82.46% because it could capture non-linear interactions and feature relationships. The architecture of Decision Tree, by recursively partitioning the data based on information gain, allowed it to deal with the dataset complexity more efficiently. In most cases, the Decision Tree performed better than the other two models and is therefore a good model for this classification task compared to KNN and Logistic Regression.

# Personal Data Selection #2 - Advertising

## Dataset Background

The data has 200 rows and four columns, three of which are features: TV, Radio, and Newspaper, representing the advertisement spending on each medium. The target feature is Sales, which is the sales result corresponding to the respective advertisement spending. The data is meant to analyze the correlation between the spending done on different forms of advertisement and the resulting sales values, giving information about the effectiveness of these advertisement media in creating sales.

## Linear & Logistic Regressions

The process begins by importing the necessary libraries and defining two custom classes: Linear Regression and Logistic Regression. Each class includes methods for training (fit), prediction (predict), and evaluation. The dataset, which includes advertising costs in TV, Radio, and Newspaper as features, and sales as the target variable, is loaded using Pandas. Features are normalized for consistency, and the target variable is transformed into a binary class for logistic regression (sales above or below the median).

Next, the data is manually split into training and test sets, ensuring that 80% of the data is used for training and 20% for testing. The Linear Regression model is trained using the training data, and its performance is evaluated using Mean Squared Error (MSE), which quantifies the average squared difference between the predicted and actual sales. The Logistic Regression model, on the other hand, uses the transformed binary target variable and is evaluated using accuracy, which calculates the percentage of correctly predicted sales classifications.

## Decision Trees

. The data is normalized and split between train and test at 80-20. The decision tree is constructed by repeatedly selecting the best feature and threshold value that minimizes Gini impurity. Construction stops once the tree is at a specified depth or when the further splits do not improve significantly. During training, the model predicts the test set sales. Model performance is determined through R-squared metric, indicating model prediction quality versus actual sales figures. Overfitting is restricted by capping the decision tree depth to a maximum of 5 with the minimum split size set at 10. There is, thereby, a trade-off achieved in terms of prediction accuracy against complexity in the model.

## KNN

The data is split into training and test sets in an 80-20 ratio. The KNN algorithm calculates the Euclidean distance between the test samples and the training samples, determines the k nearest neighbors, and makes the target value (Sales) prediction as the average of the target values of the closest neighbors.

The model's performance is quantified in terms of the R-squared value, which computes the proportion of variance in the target variable that is explained by the model. With different values of k tried, the performance of the KNN algorithm can be ascertained, with k=3 yielding a specific R² value. This provides a sense of how well the model forecasts sales given the input features.

## Bayesian Classifier

The `BayesianClassifier` class applies the principles of the Naive Bayes algorithm, assuming that features of each class are normally distributed with separate distributions. The `fit()` method calculates the mean, variance, and prior probability of each class from the training data. These are utilized to calculate the probability of each class for a sample from the test set using the input features. The classifier estimates the posterior probabilities for each class according to Bayes' theorem, and the most likely class is chosen as the predicted label. The `accuracy()` function computes the classification accuracy by comparing the predicted labels with the actual labels in the test set.

For the data set, including advertising data, the `Sales` variable is converted to a binary class if sales are above or below the median. The features (`TV`, `Radio`, `Newspaper`) are scaled to ensure that the input is in comparable scale, improving the performance of the model. The data set is divided into a training set and test set in an 80-20 ratio, and the training data is used to train the classifier. The performance of the model after training is then compared with the test set to assess its ability to classify new, unseen data.

## Discussion & Accuracy Comparison

The results indicate that the Bayesian Classifier had a high accuracy of 0.9, which means that it was effective in classifying the binary sales data. This indicates that the Naive Bayes model's assumptions, specifically the independence of features and their Gaussian distributions, were well-suited for the given dataset. However, it is to be observed that Naive Bayes would perform badly if the attributes are not independently distributed or in the event of the Gaussian assumption violation. The Linear Regression model, on the other hand, yielded a mean squared error (MSE) of 3.18, reflecting a relatively poor fit compared to other models' accuracy.

This means that Linear Regression would maybe not be the most appropriate tool for the task, especially with its linear assumptions that maybe don't fit the nonlinear relationships in the data. Logistic Regression, which was 0.925 accurate, had the same relative performance as the Bayesian Classifier and is very appropriate for the task of binary classification, employing the logistic function to model the relationship between features and probability of an outcome. The 0.876 accuracy of Decision Tree indicates that it too performed well, though perhaps more prone to overfitting than models like Logistic Regression. Finally, the k=3 KNN model had surprisingly low accuracy of 0.267, likely due to the fact that the model is sensitive to noise and is plagued by the curse of dimensionality, especially without feature scaling as well as a proper k value. Therefore, how well the model performed depended mainly on the success of the algorithm in identifying patterns within the data.

# Conclusion & Future Implementation

This was a comprehensive project on fundamental machine learning algorithms, implemented from scratch without using pre-built libraries. The models developed—Linear and Logistic Regression, Decision Trees, K-Nearest Neighbors (KNN), and Naïve Bayes classifiers—were implemented on diverse datasets, including medical diagnosis, food categorization, robotics, obesity prediction, and ad effectiveness. It was noted that model performance varied with respect to dataset characteristics. Logistic Regression did well in binary classification tasks like Breast Cancer diagnosis, while Decision Trees did well but were overfitting sensitive. KNN did well on structured data but was inefficient computationally on big data. Naïve Bayes did well on categorical data but didn't do well when features were not independent, and Linear Regression did best on regression tasks but wasn't able to model non-linear dependencies. Future improvements can involve model tuning with regularization techniques such as L1/L2 for regression, hyperparameter tuning for KNN and Decision Trees, and more complex feature engineering for improved accuracy.

Scalability problems in KNN can be resolved with efficient data structures, and pruning can prevent overfitting in Decision Trees. Employing neural networks can handle complex feature relationships where traditional models cannot perform. This project highlights the importance of learning machine learning from scratch, and sets a strong foundation for more advanced applications of predictive modeling and data-driven decision-making.

# Other

Breast Cancer data - [Breast Cancer Wisconsin (Diagnostic) - UCI Machine Learning Repository](https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic)

Robot Execution Failures - [Robot Execution Failures - UCI Machine Learning Repository](https://archive.ics.uci.edu/dataset/138/robot+execution+failures)

Mushroom - [Mushroom - UCI Machine Learning Repository](https://archive.ics.uci.edu/dataset/73/mushroom)

Obesity Levels - [Obesity Levels](https://www.kaggle.com/datasets/fatemehmehrparvar/obesity-levels)

Advertising - [Advertising dataset](https://www.kaggle.com/datasets/tawfikelmetwally/advertising-dataset)