

Course Code: ENGR 5005

Course Name: Machine Learning for Engineering Applications

Assignment: Coding Project

Prepared By: Volkan Turgut

Student ID: 100995573

Date: 27/03/2025

Table of Contents

[**Abstract** 3](#_Toc194160549)

[**Mathematical Background** 4](#_Toc194160550)

[**Linear Regression** 4](#_Toc194160551)

[**Logistic Regression** 5](#_Toc194160552)

[**Naïve Bayes Classifier** 6](#_Toc194160553)

[**K-Nearest Neighbours Classifiers** 7](#_Toc194160554)

[**Decision Tree** 7](#_Toc194160555)

[**Datasets Explanation, Comparison and Analysis** 9](#_Toc194160556)

[**Breast Cancer Dataset** 9](#_Toc194160557)

[Quick Information 9](#_Toc194160558)

[Data Preprocessing 10](#_Toc194160559)

[Results 10](#_Toc194160560)

[Analysis and Comparisons 10](#_Toc194160561)

[**Mushroom Dataset** 12](#_Toc194160562)

[Quick Information 12](#_Toc194160563)

[Data Preprocessing 13](#_Toc194160564)

[Results 13](#_Toc194160565)

[Analysis and Comparisons 14](#_Toc194160566)

[**Robot Execution Failures Dataset** 15](#_Toc194160567)

[Quick Information 15](#_Toc194160568)

[Data Preprocessing 16](#_Toc194160569)

[Results 16](#_Toc194160570)

[Analysis and Comparisons 17](#_Toc194160571)

[**Car Evaluation Dataset** 19](#_Toc194160572)

[Quick Information 19](#_Toc194160573)

[Data Preprocessing 20](#_Toc194160574)

[Results 20](#_Toc194160575)

[Analysis and Comparisons 20](#_Toc194160576)

[**Spambase Dataset** 22](#_Toc194160577)

[Quick Information 22](#_Toc194160578)

[Data Preprocessing 23](#_Toc194160579)

[Results 23](#_Toc194160580)

[Analysis and Comparisons 24](#_Toc194160581)

[**Conclusion** 25](#_Toc194160582)

[**References** 27](#_Toc194160583)

# **Abstract**

This project investigates the basic notions and implementations of main machine learning algorithms like Logistic Regression / Linear Regression / Decision Trees / K-Nearest Neighbours / Naive Bayes Classifiers / KNN. The project aims to understand how machine learning models are built, trained and tested without external machine learning libraries by focusing on their mathematical foundations.

The report reviews the theoretical underpinnings of the models and then the handling of individual datasets. Evaluation of models performance under various conditions was performed on three benchmark datasets from the UCI Machine Learning Repository and two self-selected datasets.

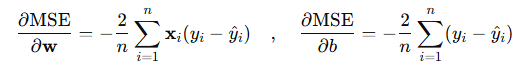
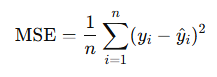
A major part of the project involves model performance comparison against accuracy and error metrics. The discussion reviews strengths and weaknesses across models and datasets to reveal their performance and limitations in different scenarios. This project avoids high-level libraries and implements the algorithms from scratch allowing a practical introduction to machine learning mechanics and its real world application.

# **Mathematical Background**

## **Linear Regression**

Linear Regression models the relationship between input features and a continuous output variable. It assumes a linear relationship defined by:



Where X is the feature vector, w is the weight vector, b is the bias and y\_hat is the predicted output. The model’s objective is to minimize the error function which is chosen to be MSE (mean squared error). Gradient descent is used to adjust the weights and bias by taking derivative of the loss function to make sure that we are on the path to decrease MSE each time we take a step towards to optimal weight and bias values with gradient descent.

The gradient descent of weight is multiplied by learning rate and deducted from the present weight values. Same goes for bias, gradient descent is calculated, multiplied by learning rate and it is subtracted from the present bias value. Though designed for continuous labels, Linear Regression is not suitable for this project since all datasets have categorical output classes. In this context MSE is no meaningful performance metric. Still evaluation of model was done by rounding predicted outputs to the nearest integer for class predictions simulation and success rate (Accuracy) computing.

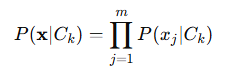
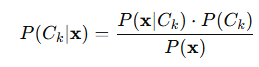
## **Logistic Regression**

Logistic Regression builds on Linear Regression but transforms its output using the **sigmoid function** to map predictions into the range suitable for **classification.**



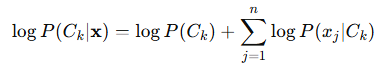
This lets the model interpret the output as probability and assign class labels depending on a threshold (usually 0.5). Logistic Regression is particularly useful with categorical target variable because it attempts to fit decision boundary between classes by smooth curve.

## **Naïve Bayes Classifier**

A probabilistic model based on Bayes' Theorem called Naive Bayes assumes features are conditionally independent from the class label. For input x, for class Ck the probability is:

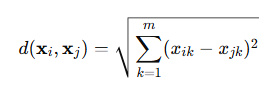
Knowing that x actually represents a set of probability, by choosing a class Ck, we can actually compute the probability of that specific class.

Since what we are looking for is argmax of k where the probability of class Ck given that x, the left hand side of the equation becomes as above. P(x) doesn’t ruin the proportional relations between the probabilities so it can be discarded from the equation as the argmax k value won’t change. Computing the set of chain multiplications is quite hard, so we take the logarithm of the whole equation leaving us with the equation below:



The probability of class Ck happening is called log prior, the logarithm of previously multiplication, now logarithm addition part is called log likelihood, and the right hand side is the log posterior. Our aim is to find all of the log posterior’s, compare them with each other and take the maximum as it yields a bigger value in probability of class Ck given the x input, that will be our prediction.

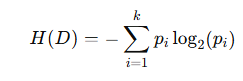
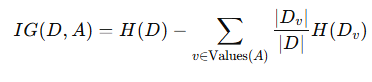
## **K-Nearest Neighbours Classifiers**

KNN is a non-parametric method. Given a test instance x, it finds the k closest training examples using a distance metric (usually Euclidean Distance):

The Euclidian distances are compared and the least k ones have been selected. Among these k closest neighbours, the majority is chosen for the prediction value. KNN doesn’t need to be trained, as it is actually a run time trained model, however there are lots of calculations involved, so this will be one of the most time consuming ML model’s I have ran today.

## **Decision Tree**

Decision Trees partition the data by recursively splitting it based on features that provide the highest **Information Gain (IG)**. The most common splitting criteria include **Entropy** and **Gini Index**.



Entropy and information gain formula has been given respectively above. In this project, splitting criterion was Entropy. For categorical features entropy is computed per possible value. For features with multiple categories, each subset is weighted by its size and their entropies are summed up and subtracted from the overall dataset entropy.

For numerical features the data are sorted and potential split points occur between consecutive values. With n unique values there can be up to n-1 splits at most assuming there are no overlapping feature values. For each split: the data is sorted into two subsets from the split point, left and right subset entropies are computed, respectively, they are added to their weighted sum to get IG minus total entropy, this process is repeated for all split points within the feature, and the maximum IG represents that feature’s (column’s) information gain. Then these columns are compared with each other, and the feature (column) that yields the highest information gain is chosen to be the split decision and branched. This process is done recursively until either all labels in a branch are the same, or the maximum depth is reached, or the remaining samples are smaller than 5 which majority of the label’s create the leaf node in that case.

# **Datasets Explanation, Comparison and Analysis**

## **Breast Cancer Dataset**

### Quick Information

The characteristics of a digitized image of a fine needle aspirate (FNA) of a breast mass are described from breast cancer biopsies and are used to classify tumors as malignant (M) or benign (B). There are 569 Instances in the dataset.

No. of features: 30 real value Features.

Diagnosis (M for malignant, B for benign):

Identifier Column: Present, but not used for modeling.

Features: Radius, Texture, Perimeter, Area, Smoothness, Compactness, Concavity, Concave, points, Symmetry, Fractal dimension

Thus, 10 measurement types x 3 groups = 30 features. All of the features are continuous values.

Missing Values:

It has no missing values. The records contain all the 30 feature values together with a diagnosis label.

Label Distribution:

Malignant (M): 212 samples

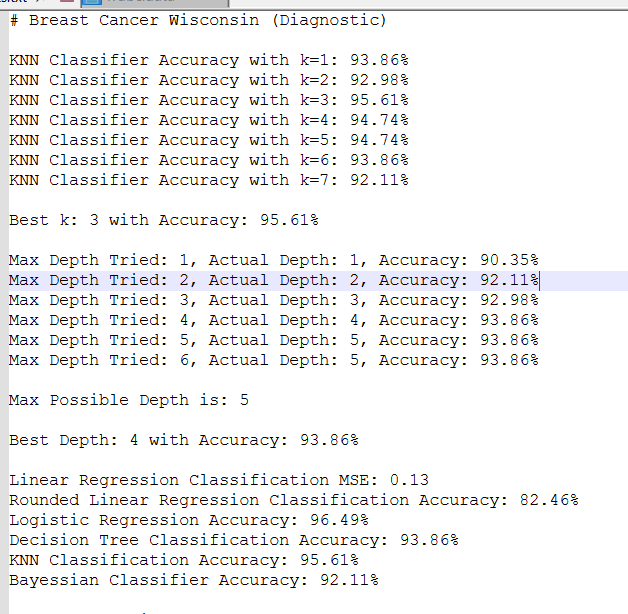
Benign (B): 357 samples

Benign cases are more frequent in the dataset.

### Data Preprocessing

ID column is dropped. Then the non-numerical categorical columns were mapped into integers starting from 0. Then continuous columns were normalized, otherwise due to the nature of columns spanning different ranges of numbers, it causes the model to be biased. Thus continuous columns were normalized. Then dataset was split into input and output, and then finally, dataset has been shuffled and separated into 80-20 ratio training and testing.

### Results



### Analysis and Comparisons

**Linear regression** is actually not suitable for classification tasks however, for the sake of completing the project it was implemented. This produced a Mean Squared Error (MSE) of 0.13 and classification accuracy of 82.46% when continuous predictions were rounded to the nearest integer. This shows some predictive capacity but also demonstrates the limitations of a regression model for categorical output problem. As the dataset labels are class based, MSE isn't a useful performance metric by itself and the relatively low accuracy is due to the model's incapacity to draw effective class boundaries.

**Logistic Regression** performed best with 96.49% accuracy compared to all models. Such a strong result is expected, since the dataset is clean, linear to a degree separable and binary, in keeping with Logistic Regression strengths. By applying the linear model with sigmoid function it maps input data into class probabilities and distinguishes the malignant and benign cases with high reliability. Its performances show it can handle such datasets well.

Best accuracy for **KNN** was 95.61% when k = 3. Lower values of k performed better; accuracy peaked at 3 and gradually decreased for higher values; this was probably due to over fitting and loss of sensitiveness of the model. Its strength is its simplicity, its ability to learn the distribution of data without any training, but the efficacy depends on the optimal k. In this case k = 3 was ideal and the results were almost as strong as Logistic Regression.

The maximum accuracy of **Decision tree** classification was 93.86% when the Tree depth was set to 4. Performance had increased with depth to this point and then stalled, suggesting deeper trees did consume more computational power and were also potentially overfitting. The last tree reached 5 feet deep. These results indicate that a fairly shallow tree suffices for the dataset and that interpretable & capable Decision Trees may be interpretable but not precise enough for this task as more stable models such as Logistic Regression and KNN.

**Naive Bayes** reached 92.11% accuracy - less than the other models but still respectable given its simplicity. The model assumes that all features are independent from the class label, something that rarely happens in real data. The 30 features of this dataset are probably correlated, so Naive Bayes might have performed poorly. But because of its fast training time and solid baseline performance it remains a good model for initial exploration and short deployment scenarios.

## **Mushroom Dataset**

### Quick Information

The Mushroom dataset is a set of hypothetical samples for 23 gilled Mushroom species of the Agaricus/Lepiot family. Various categorical attributes include cap shape, color, gill size, odor, habitat and spore print color. It is aimed at classifying each mushroom as either edible e) or poisonous p) according to their physical characteristics. Its rich set of categorical features and balanced label distribution make the dataset widely used for classification tasks. There are 8124 such instances.

No. of features 22 categorical attributes - not including class label, class Label: e' for edible, p' for poisonous.

One character per feature represents a categorical value. It contains no numerical or continuous features. All columns are non-numeric and must be encoded before modeling.

**Missing Values:**

Some of the features have ‘?’ because they are missing. Those values needs to be updated.

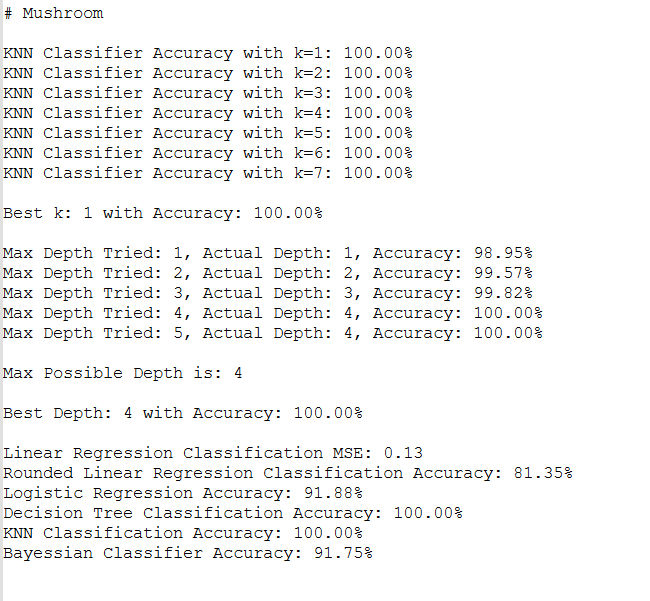
Label Distribution | Edible (e): 4208 samples / Poisonous (p): 3916 samples

This almost balanced dataset is suitable for objective evaluation of classification model performance without bias.

### Data Preprocessing

Missing values replaced (this dataset has ‘?’ in some spots so these need to be filled with ‘0’ mock values). Then the non-numerical categorical columns were mapped into integers starting from 0. No normalization needed for this dataset because there are no continuous values. Then dataset was split into input and output, and then finally, dataset has been shuffled and separated into 80-20 ratio training and testing.

### Results



### Analysis and Comparisons

**Linear regression** was tested again to see how it performs in such clearly categorical dataset. The Mean Squared Error (MSE) was 0.13 and the rounded classification accuracy was 81.35%. This relatively low accuracy demonstrates that regression is inappropriate for such classification tasks. Since the features are completely categorical with discrete classes labels, the regression model cannot draw meaningful bounds. Output was rounded to mimic classification but the performance shows its ineffectiveness in this domain.

**Logistic Regression** was 91.88% accurate despite the dataset being entirely categorical. Transforming input via sigmoid function gave the model the ability to estimate probabilities and class labels but not to its full potential due to lack of numerical relationships. Nevertheless, it was able to generalize patterns in the data with respectable accuracy.

**KNN** predicted 100% accuracy for all tested values k of 1 - 7. It thus confirms that the dataset is well structured and suitable for KNN since similar instances are very close in the encoded feature space. No learning or training phase was required for the algorithm to recognize edible and poisonous mushrooms. This actually shows that probably, there is a perfect correlation between a single or couple of the features which makes the data highly seperable.

**Decision Tree** classification also gave 100% accuracy to the maximum depth of 4. Accuracy increased incrementally to 98.95% from depth 1 to perfect classification below which no further improvements were recorded. The relatively shallow depth required suggests that a small subset of features contains highly discriminative information for classification task. This shows that the decision tree is able to handle categorical data and find key splits well without overfitting.

**Naive Bayes** returned 91.75% accuracy, slightly lower than Logistic Regression but quite strong. Its performance is good considering its simplistic assumptions of feature independence. The model was effective despite possible correlations between features like cap color and gil color. The efficiency of its training process and relatively fast inference make it a reliable choice for fast classification tasks but here it clearly outperformed tree-based and distance-based models.

## **Robot Execution Failures Dataset**

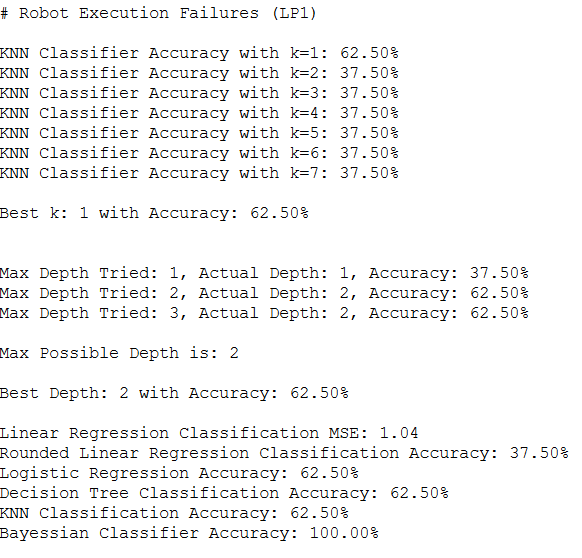
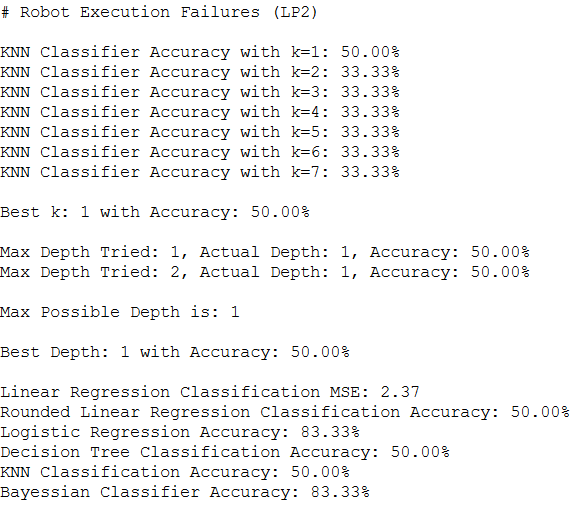
### Quick Information

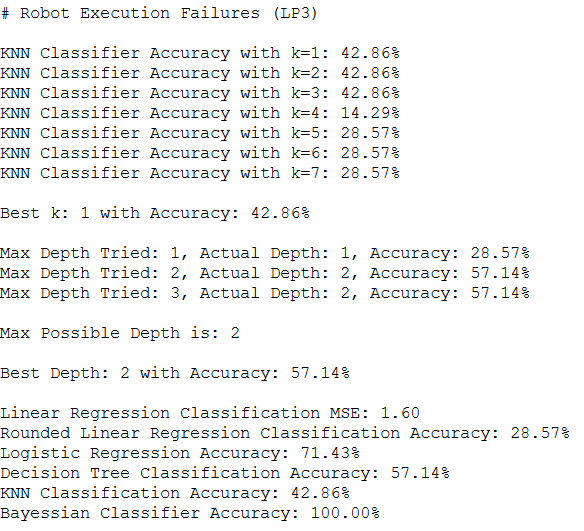
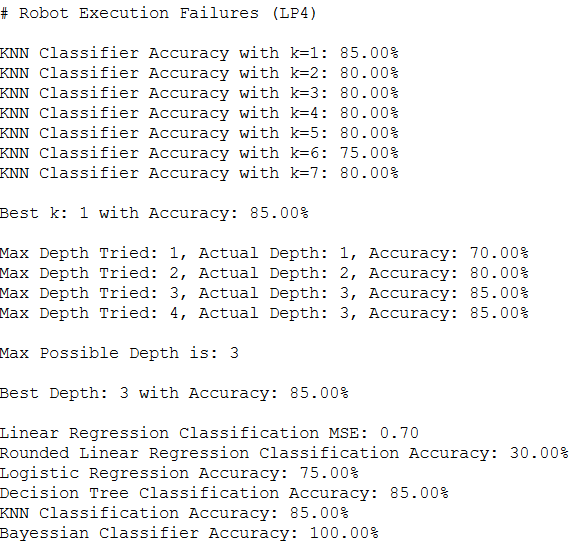
A multivariate time series Robot Execution Failures dataset containing five learning problems (LP1 - LP5) representing five different robotic failure types. Every sample represents one failure instance and contains 90 numerical features derived from 15 time-ordered measurements of force and torque in six dimensions (Fx, Fy, Fz, Tx, Ty, Tz). These measurements are acquired shortly after the failure detection in a time window of 315 milliseconds. Almost every learning problem represents a robotic context like grasp failure, ungrasp failure, part motion failure, etc. All features are integer-valued and no datasets are missing values. Almost all datasets are small: LP3, LP2, and LP1 have under 90 instances, LP4 and LP5 have over 100 instances. The class labels differ across datasets where some imbalance exists, and each dataset was treated independently in this project. Due to the uniform scale, all features were used as is without normalization and the data were shuffled into 80% training/20% testing sets.

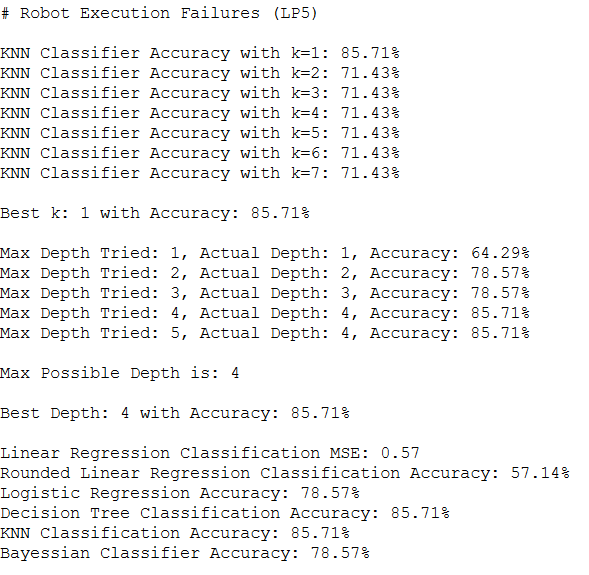
### Data Preprocessing

The non-numerical categorical columns were mapped into integers starting from 0. No normalization needed for this dataset because there are no continuous values. Then dataset was split into input and output, and then finally, dataset has been shuffled and separated into 80-20 ratio training and testing.

### Results







### Analysis and Comparisons

**Linear regression** performed poorly across all five robot learning problems. Its rounded classification accuracy was lowest in LP3 at 28.57%, and highest in LP2 and LP1 at 50.00% and 37.50% respectively. MSE values were consistently high, especially in LP2 where it reached 2.37. These results underline how unsuitable regression is for multi-class classification tasks involving high-dimensional, time-series data. The model’s inability to create categorical decision boundaries leads to unreliable predictions even after rounding.

**Logistic Regression** showed varying levels of performance depending on the dataset. In LP2 and LP5, it achieved the highest accuracy of 83.33% and 78.57%, respectively. LP1 and LP4 saw a moderate performance of 62.50% and 75.00%, while LP3 fell in the middle with 71.43%. These results demonstrate that logistic regression can adapt to certain data structures but may struggle when class separation becomes subtler or class imbalance is more pronounced.

**KNN** had its best performances in LP4 and LP5, reaching 85.00% and 85.71% accuracy, respectively, both at k=1k = 1k=1. LP1 followed with 62.50%. However, in LP2 and LP3, accuracy dropped to 50.00% and 42.86%, respectively. In most cases, increasing k beyond 1 led to immediate performance degradation, often stabilizing at lower accuracy. This suggests that for these specific time-series datasets, a minimal k provides better locality-based predictions, but overall performance is inconsistent due to feature complexity and dataset size.

**Decision Tree** results were identical to KNN in every learning problem, indicating that both models were equally capable of separating classes in the available feature space. LP4 and LP5 again stood out, reaching 85.00% and 85.71%, with LP1 and LP2 reaching 62.50% and 50.00%. LP3 achieved a modest 57.14%. Tree depth was relatively shallow, maxing out at 4 in LP5, and the limited growth likely helped prevent overfitting while still allowing meaningful splits.

**Naïve Bayes** delivered surprisingly strong performance and was the top performer in several cases. It achieved perfect 100.00% accuracy in LP1, LP3, and LP4, which is remarkable given the model’s independence assumptions. In LP2 and LP5, its accuracy remained high at 83.33% and 78.57%. These results suggest that the structured force/torque signals in each time step are sufficiently distinct to allow independent probability estimation per feature. Naïve Bayes outperformed or matched all other models in nearly every LP, making it the most consistent performer across the board.

## **Car Evaluation Dataset**

### Quick Information

The car Evaluation dataset contains categorical features describing Car attributes and attempts to classify cars into acceptability levels. All instances are rated unacceptable, acceptable, good or very good. It draws on a hierarchical decision model developed for assessing car quality on structural attributes such as safety, price and maintenance. There are 1728 instances.

No. of features: 6 categorical attributes

Class Labels:' unacc',' acc',' good',' vgood'.

The attributes are buying price, maintenance cost, number of doors, capacity in persons, luggage boot size and safety. All features are nominal and non-numeric, so categorical values were encoded to integer representation during preprocessing.

Missing Values:

The dataset contains no missing values. All records are complete and logically arranged for classification purposes.

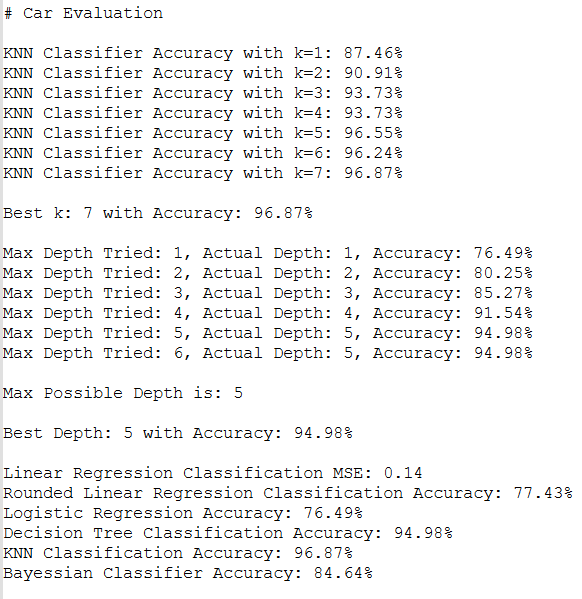
Label Distribution:

The dataset is slightly imbalanced with 'unacc' being the dominant class. But all four classes are sufficiently enriched to permit unbiased supervised learning.

### Data Preprocessing

The non-numerical categorical columns were mapped into integers starting from 0. No normalization needed for this dataset because there are no continuous values. Then dataset was split into input and output, and then finally, dataset has been shuffled and separated into 80-20 ratio training and testing.

### Results



### Analysis and Comparisons

**Linear regression** had the lowest MSE and rounded classification accuracy of all models, however. Since the dataset is entirely categorical and the target is multi-class, the regression model had difficulty learning meaningful patterns. The predicted outputs were only rounded to match the correct class in simpler cases. This again demonstrates that regression is not suited for classification tasks, at least in datasets without numerical relationships.

**Logistic regression** was similar to linear Regression with an end accuracy of 76.49%. It is a classification model whose structure is suitable for binary outcomes. Multi-class output and categorical feature space probably contributed to its limited performance. It could recognize some class separation but not model all four target categories with high confidence.

**KNN** performed best among tested models with 96.87% top accuracy with k = 7. With increasing k, accuracy improved steadily, suggesting that the model was better served by wider neighborhood consideration. This result demonstrates that KNN could group similar encoded instances and assign the right class based on local similarity in feature space. The model particularly handled the structure of this dataset better than tree-based and probabilistic approaches.

Similarly, **Decision Tree** classification performed well with 94.98% accuracy to the maximum depth of 5. The accuracy was increasing with depth, suggesting deeper splits let the tree uncover complex feature interactions. All features are categorical/discrete, so the decision tree formed very specific rules for matching class labels. Finally, the model matched depth and accuracy without overfitting.

With an 84.64% accuracy, **Naive Bayes** performed well but was behind KNN and Decision Trees. In spite of the independence assumption, the model captured sufficient feature-label relationships to classify most examples correctly. Its categorical nature worked well with Naive Bayes but some feature interactions were probably too dependent for the model to handle fully. It still showed good and efficient performance.

## **Spambase Dataset**

### Quick Information

The Spambase dataset contains email data represented as statistical features based on word frequency and character frequency. It attempts to classify emails as spam or not spam. Each instance has 57 continuous features including word frequencies / character frequency / capital letter statistics / etc. It is a dataset of general interest in spam detection research and serves as solid benchmark for binary classification models. It has 4601 instances.

Features: 57 continuous numerical attributes.

Class Labels: 0 for spam, 1 for spam.

Features include percentage frequency of specific words e.g., free, "click," money; character frequencies e.g.:' USD "#'' !'; average run length of capital letters; thus, the dataset has good descriptive power for classification tasks.

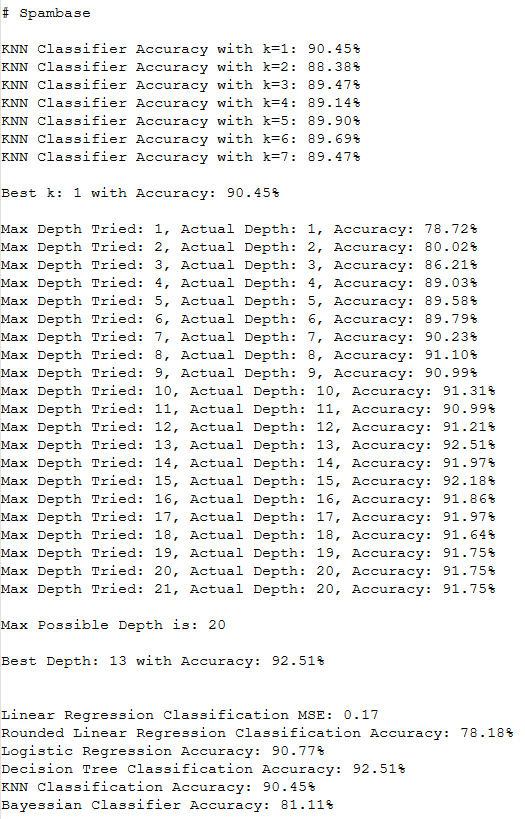
Missing Values: Even though in the <https://archive.ics.uci.edu/dataset/94/spambase> website, under the description it says there are missing values. However, in the .DOCUMENTATION file, it states that there are no missing values. The generic function for replacement for missing values block is anyway applied like in any other dataset just in case for ‘?’, and NaN values.

Label Distribution: The class labels are slightly uneven at about 60% non-spam/40% spam. Cette distribution is sufficient for balanced model training without bias handling.

### Data Preprocessing

Just in case, the replacement of missing data code block is applied because there is a mismatch between the website and .DOCUMENTATION file. Then continuous columns were normalized, otherwise due to the nature of columns spanning different ranges of numbers, it causes the model to be biased. Thus continuous columns were normalized. Then dataset was split into input and output, and then finally, dataset has been shuffled and separated into 80-20 ratio training and testing.

### Results



### Analysis and Comparisons

In this dataset, **Linear Regression** performed poorly compared with other models. It scored 78.18% rounded classification accuracy with 0.17 MSE. The model captured some trends in the data but failed to distinguish categorical classes, causing deformed results. This again demonstrates that linear regression is inappropriate for classification tasks especially with imbalanced categorical label output and continuous input features.

**Logistic Regression** was very accurate (90.77%). This fits the nature of the problem since logistic regression is explicitly tuned for categorical classification, especially binary which is the case, tasks. The model handled continuous, normalized features well and set a strong decision boundary for spam versus non-spam examples. Its performance demonstrates that simple linear classifiers perform well with prepared datasets.

**KNN** performed very well but slightly worse than some other models, best accuracy being 90.45% k = 1. With increasing k, the accuracy slipped slightly but never surpassed the initial peak. The higher k performance drop suggests that local structure was critical for prediction and that nearest neighbor alone gave a good indication of class. It was still competitive and consistent nonetheless throughout the experiment.

Among all models, **Decision Tree** classification performed best with an accuracy of 92.51% at depth 13. From 1 to 13 depths, accuracy improved slightly before plateauing. This shows that word frequency and character patterns required complex decision rules to distinguish spam from legitimate emails. The decision tree was well adapted to the data structure and showed good classification with interpretable splits.

**Naive Bayes** had an accuracy of 81.11%, the lowest of all classifiers apart from linear regression. Though still reasonably performant, its assumptions of feature independence rendered it less effective. With respect to some spam-indicative words and patterns, the model did not capture conditional relationships between features. Its fast training time and simplicity notwithstanding, it was a useful baseline.

# **Conclusion**

I built a general reusable framework for training & evaluation of machine learning models on different classification datasets. Though some datasets had missing values, I built my preprocessing pipeline generic and robust enough to handle datasets of any structure as long as the target output is categorical. This included validation and replacement for missing values, so that the code can be easily adapted to unknown or incomplete datasets without any modification.

Each dataset followed a consistent preprocessing pipeline. At first, the dataset was accessed and parsed by format. Categorical features were then mapped to numerical values and missing values were validated and replaced. If continuous features were present, they were normalized. Normalization was uniform across the pipeline even in datasets without numeric features. Finally, all datasets were shuffled into training and testing sets at 80-20 ratio. This whole flow ensured that all datasets regardless of structure went through standardized transformation before model evaluation.

The models themselves were implemented by a generic interface. Each classifier was defined to be a Linear Regression, Logistic Regression, K-Nearest Neighbours, Decision Tree or Naive Bayes function having same structure and callable execution: This modularity streamlined experimentation and showed how machine learning is scalable and maintainable if abstracted.

I learned about the core mechanisms of classical machine learning models through this project. I learned the mathematics, data handling and performance trade-offs of each approach by avoiding external libraries for training and building each model from scratch. I learned about preprocessing, how to work with imbalanced / high-dimensional data and how to select the right model for the dataset structure / type.

Also, this project illustrated how important empirical comparison is. Tests on different datasets ranging from medical diagnostics to spam filtering to robot failure detection showed me firsthand how model performance can vary significantly across data characteristics. Logistic Regression & Decision Trees generally performed well on structured data whereas Naive Bayes performed well on clean, high dimensional datasets and where features where highly independent. KNN was effective in some cases but sensitive to parameter tuning and to data distribution.

This project has essentially taught me to code/debug and also to conceptualize machine learning. It taught me to critically think about the whole ML pipeline: from raw data to educated evaluation - and to build mathematically sound, practically usable end-to-end solutions. I could improve the implementation to support datasets having continuous output values by adding regression focused models and loss functions. This would let the framework handle classification and regression problems and make it generic and general enough to handle a larger variety of machine learning tasks. Other useful evaluation metrics for regression could be or MAE.

# **References**

[1] D. Dua and C. Graff, “UCI Machine Learning Repository: Spambase Data Set,” University of California, Irvine, School of Information and Computer Sciences. [Online]. Available: <https://archive.ics.uci.edu/dataset/94/spambase>

[2] D. Dua and C. Graff, “UCI Machine Learning Repository: Car Evaluation Data Set,” University of California, Irvine, School of Information and Computer Sciences. [Online]. Available: <https://archive.ics.uci.edu/dataset/19/car+evaluation>

[3] D. Dua and C. Graff, “UCI Machine Learning Repository: Breast Cancer Wisconsin (Diagnostic) Data Set,” University of California, Irvine, School of Information and Computer Sciences. [Online]. Available: <https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic>

[4] D. Dua and C. Graff, “UCI Machine Learning Repository: Robot Execution Failures Data Set,” University of California, Irvine, School of Information and Computer Sciences. [Online]. Available: <https://archive.ics.uci.edu/dataset/138/robot+execution+failures>

[5] D. Dua and C. Graff, “UCI Machine Learning Repository: Mushroom Data Set,” University of California, Irvine, School of Information and Computer Sciences. [Online]. Available: <https://archive.ics.uci.edu/dataset/73/mushroom>