MINOR PROJECT

OF

SUMMER TRAINING, UNDERTAKEN

AT

Novem Controls

ON

MACHINE LEARNING

SUBMITTED IN PARTIAL FULFILLMENT OF THE DEGREE

OF

BACHELOR OF TECHNOLOGY

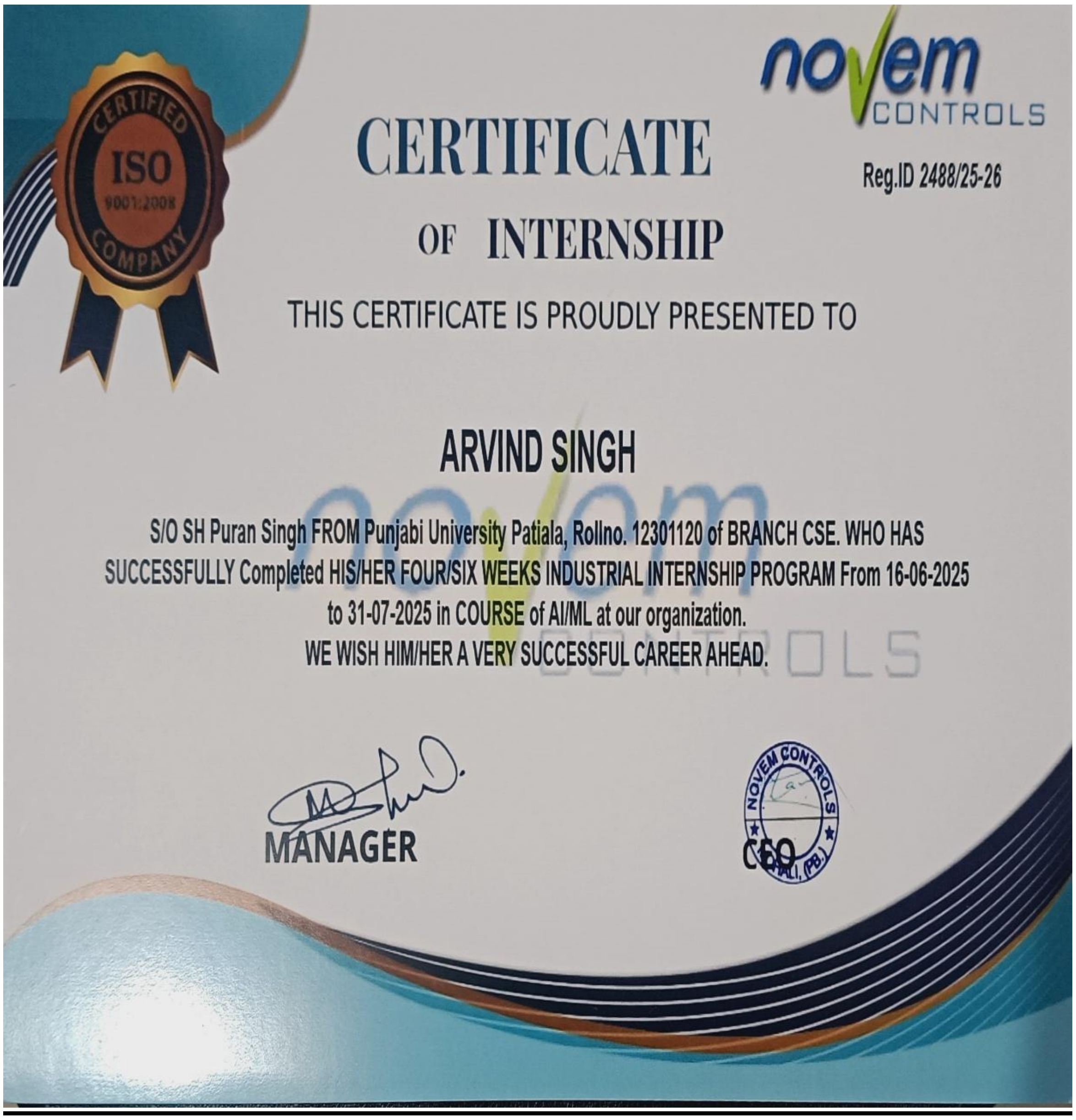
IN

COMPUTER SCIENCE AND ENGINEERING

Mentored By: Dr. Gurjit Singh Bhathal

Submitted By:
Name: Arvind Singh
Roll No:12301120

CERTIFICATE BY THE INSTITUTION



novem CONTROLS

CERTIFIED ISO 9001:2008 COMPANY

Reg.ID 2488/25-26

CERTIFICATE OF INTERNSHIP

THIS CERTIFICATE IS PROUDLY PRESENTED TO

ARVIND SINGH

S/O SH Puran Singh FROM Punjabi University Patiala, Rollno. 12301120 of BRANCH CSE. WHO HAS SUCCESSFULLY Completed HIS/HER FOUR/SIX WEEKS INDUSTRIAL INTERNSHIP PROGRAM From 16-06-2025 to 31-07-2025 in COURSE of AI/ML at our organization.

WE WISH HIM/HER A VERY SUCCESSFUL CAREER AHEAD.

[illegible]

MANAGER

NOVEM CONTROLS MOHALI (PB.)

CEO

CANDIDATE’S DECLARATION

I Arvind Singh hereby declare that I have undertaken Summer training at NOVEM CONTROLS during the period of 16 June to 31 July in partial fulfillment of requirements for the award of degree of B.Tech (Department of Computer Science & Engineering) at Punjabi University Patiala. The work which is being presented in the training report submitted to the Department of Computer Science & Engineering at Punjabi University, Patiala is an authentic record of training work.

Signature of the Student

The summer training Viva-Voce Examination of ___________________________ has been held on ____________________ and accepted.

Signature of the Examiner

ABSTRACT

Mushroom Edibility Classification using Logistic Regression: A Performance Analysis

The accurate and rapid classification of mushrooms into edible and poisonous categories is a crucial task for food safety and mycology. Misidentification can lead to severe health issues or even death due to the toxic nature of many species. This study focuses on developing a predictive model to classify mushroom edibility based on their morphological characteristics using a machine learning approach.

The dataset, comprising **8124 instances** and **22 categorical features** describing various mushroom attributes (e.g., cap shape, odor, gill color, habitat), was preprocessed for model training. This involved separating the target variable ('class' - 'p' for poisonous, 'e' for edible) and applying **Label Encoding** to the target. The predictor variables underwent **feature encoding**, utilizing **One-Hot Encoding** for features assumed to be nominal and **Ordinal Encoding** for those assumed to have an inherent order. The preprocessed data was then scaled using **StandardScaler (with with_mean=False)**.

A **Logistic Regression** model was selected for the binary classification task and trained on a 80/20 train-test split of the processed data, using a **stratified sampling** technique to maintain class balance. Model performance was evaluated using standard metrics on the test set. The model achieved a high **accuracy of 0.969**, a **precision of 0.958**, and a **recall of 0.98** for the poisonous class (Class 1, 'p'), as indicated by the classification report and confusion matrix. The results demonstrate that the Logistic Regression model effectively distinguishes between edible and poisonous mushrooms based on their provided attributes, providing a reliable tool for classification.

ACKNOWLEDGEMENT

I am deeply grateful for the support received throughout the successful completion of my project, **Mushroom Edibility Classification using Logistic Regression**.

My sincere thanks go to **Novem Controls** for the opportunity to participate in their **AI/ML** program, which provided the essential foundation for this work.

I am sincerely thankful to **Dr. Gurjit Singh Bhathal,** my project guide, for his continous support, guidance , and valuable suggestions throughout the duration of this project. His constant motivation and expert insights have been instrumental in the successful completion of the work.

I extend special appreciation to my teacher, **Vikram Rajpurohit**, whose expert guidance, valuable insights, and continuous encouragement were instrumental to the project's success.I am thankful to the report advisory committee for their thoughtful feedback and suggestions, which were crucial in refining the website's functionalities and design. Additionally, I want to acknowledge the dedicated staff at **Novem Controls** for fostering a positive and productive learning environment.

I also acknowledge the **Department of Computer Science** at **Punjabi University, Patiala** for their support, which enabled me to balance my academic commitments with this practical learning experience.

Finally, I extend my deepest gratitude to my family for their constant support and encouragement. Their belief in my abilities has been a significant motivator throughout this project.

In conclusion, I am thankful to everyone who has been a part of this journey. The skills and knowledge gained from this project have been invaluable and will serve as a strong foundation for my future endeavors in **Machine Learning and Artificial Intelligence.**

Arvind Singh

INTRODUCTION TO ORGANIZATION

novem CONTROLS
An ISO 9001 : 2008 Certified Company

Novem Controls is a Top-rated IT Company based in India serving global clients since 15+ years. Over the years, They have established themselves as acclaimed global digital service provider for Consultation, Design, Development & Marketing of Web and Mobile based applications Manufacturing, trading, supplying and providing services of a premium quality range of Distributed Control System, Programmable Logic Controller, SCADA System, AC Drive Variable Frequency Drive, Pressure Instrument, Level Measurement, Temperature Instrument, Flow Instrument, etc.

Novem Controls was established in 1999 and are an ISO 9001:2008 certified company. They are a team of technocrats and professionals providing Website Development, App Development, Digital Marketing, and ERP Development. They are the foremost Manufacturer, Trader, Supplier and Service Provider of Actuators, Programmable Logic Controller, Distributed Control System, Online Water Analyzer.

LIST OF FIGURES

CONTENTS

CHAPTER 1 - INTRODUCTION

1.1 Python

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. Developed by Guido van Rossum and first released in 1991, Python emphasizes code clarity and allows programmers to express concepts in fewer lines of code compared to other languages. It supports multiple programming paradigms, including object-oriented, functional, and procedural approaches. Python's extensive standard library and active community make it suitable for a wide range of applications such as web development, data analysis, artificial intelligence, automation, and scientific computing. Its cross-platform compatibility and integration capabilities with other languages and tools further enhance its usability. Due to its easy-to-learn syntax and powerful features, Python has become one of the most popular programming languages in education, research, and industry worldwide.

1.2 Numpy In Python

NumPy (Numerical Python) is a powerful open-source library in Python that provides support for large, multi-dimensional arrays and matrices, along with a wide range of mathematical functions to operate on them efficiently. It is the foundation for many scientific computing and data analysis libraries in Python, such as pandas, SciPy, and scikit-learn. NumPy's array objects (ndarrays) enable faster computation compared to traditional Python lists by using optimized C-based operations. The library also includes functionalities for linear algebra, Fourier transforms, random number generation, and advanced broadcasting techniques. Due to its speed and flexibility, NumPy is widely used in data science, machine learning, and numerical simulations. It serves as a core tool for researchers and developers to perform efficient numerical computations and handle large datasets with ease.

Installation : NumPy can be easily installed using Python's package manager, pip. Openyour command prompt or terminal and run:

```
bash

pip install numpy
```

Once installed, it can be imported into Python using:

```
python

import numpy as np
```

1.3 Pandas in Python

Pandas is a powerful open-source data analysis and manipulation library built on top of NumPy. It provides flexible and efficient data structures like **Series** (one-dimensional) and **DataFrame** (two-dimensional) that allow users to handle structured data easily. Pandas simplifies tasks such as data cleaning, transformation, analysis, and visualization by offering a wide range of built-in functions. It supports importing and exporting data from various formats including CSV, Excel, SQL databases, and JSON. With its high performance and ease of use, Pandas is widely used in data science, machine learning, and business analytics for managing large datasets.

Installation: Pandas can be installed by the following command:

```
bash

pip install pandas
```

It can be imported using:

python

```
import pandas as pd
```

1.4 MatplotLib in Python

Matplotlib is a widely used open-source library in Python for creating static, interactive, and animated visualizations. It provides an easy way to plot data in various forms such as line charts, bar graphs, histograms, scatter plots, and pie charts. Built on NumPy, Matplotlib allows users to visualize data efficiently and customize every element of a graph — from colors and labels to styles and axes. It serves as the foundation for other advanced visualization libraries like **Seaborn** and **Plotly**. With its integration capabilities in Jupyter Notebooks and other Python environments, Matplotlib is essential for data analysis, research, and presentation.

It can be installed using the following command:

bash

```
pip install matplotlib
```

After installation, it can be imported using:

python

```
import matplotlib.pyplot as plt
```

1.5 Machine Learning

Machine Learning (ML) is a branch of artificial intelligence (AI) that enables computer systems to learn from data and improve their performance without being explicitly programmed. It focuses on developing algorithms that can automatically identify patterns, make predictions, and adapt their behavior based on experience. The core idea behind machine learning is that systems can learn from data — recognizing structures and relationships — to make informed decisions or predictions when exposed to new, unseen information.

Machine learning can be broadly categorized into three main types: **supervised learning**, **unsupervised learning**, and **reinforcement learning**. In supervised learning, the model is trained on labeled data where the input and corresponding output are known, allowing it to predict outcomes for new data accurately. Common algorithms include Linear Regression, Decision Trees, and Support Vector Machines. Unsupervised learning, on the other hand, deals with unlabeled data and aims to discover hidden patterns or groupings within it, such as through clustering or dimensionality reduction techniques like K-Means and PCA. Reinforcement learning involves an agent interacting with an environment to learn optimal strategies through trial and error, commonly used in robotics and gaming applications.

Machine learning relies heavily on mathematical concepts such as statistics, probability, and linear algebra, as well as computational tools for handling and processing large datasets. Python has become the most popular language for implementing ML algorithms, thanks to libraries like **scikit-learn**, **TensorFlow**, **PyTorch**, and **Keras**, which simplify model development and deployment.

The applications of machine learning span across numerous fields — from predictive analytics and recommendation systems to natural language processing, computer vision, healthcare diagnostics, and autonomous systems. For instance, ML powers spam filters in emails, fraud detection in banking, personalized content recommendations on streaming platforms, and speech recognition in virtual assistants.

In conclusion, machine learning has revolutionized how data is utilized in the modern world. By enabling systems to learn and evolve autonomously, it continues to play a pivotal role in shaping future technologies and driving innovations across industries.

1.6 Algorithms in Machine Learning

Supervised Learning Algorithms

Supervised learning models are trained on **labeled data** to predict an output variable (**y)** from the input data (**X**)

Regression Algorithms (Predicting Continuous Values)

1.1.1 Linear Regression

This algorithm models the relationship between a continuous dependent variable (y) and one or more independent variables (x) by fitting a straight line or plane (hyperplane) to the data. It determines the best-fitting line by finding the coefficients (β) that **minimize the sum of the squared vertical distances** from each data point to the line, known as the **Residual Sum of Squares (RSS).**

$$\min_{\beta} \mathrm{RSS}(\beta) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{N} (y_i - (\beta_0 + \beta^T \mathbf{x}_i))^2$$

1.1.2 Ridge Regression (L2 Regularization)

An extension of Linear Regression that addresses **multicollinearity** (highly correlated features) and prevents **overfitting**. It does this by adding an **L2 penalty term** (the sum of the squared magnitude of the coefficients, multiplied by a regularization parameter (λ) to the standard RSS objective function. This penalty **shrinks the coefficients** towards zero but rarely sets them exactly to zero.

$$\min_{\beta} \left(\sum_{i=1}^{N} (y_i - \beta^T \mathbf{x}_i)^2 + \lambda \sum_{j=1}^{P} \beta_j^2 \right)$$

1.1.3 Lasso Regression (L1 Regularization)

Also an extension of Linear Regression, Lasso (Least Absolute Shrinkage and Selection Operator) adds an **L1 penalty term** (the sum of the absolute magnitude of the coefficients, multiplied by λ) to the RSS. A key feature of Lasso is that it can force some coefficient estimates to be **exactly zero**, thereby performing **automatic feature selection** and creating simpler, more interpretable models.

$$\min_{\beta}\left(\sum_{i=1}^{N}(y_i-\beta^T\mathbf{x}_i)^2+\lambda\sum_{j=1}^{P}|\beta_j|\right)$$

Classification Algorithms (Predicting Discrete Categories)

1.1.4 Logistic Regression

A **linear classification** algorithm used to estimate the probability of a categorical outcome (e.g., binary: Yes/No, 0/1). It applies the **Sigmoid (or Logistic) function** to the linear combination of inputs to map the output to a probability value between 0 and 1. This probability is then thresholded (usually at 0.5) to assign a class.

Mathematical Formula (Sigmoid Function):

$$P(y=1|\mathbf{x})=\frac{1}{1+e^{-(\beta_0+\beta^T\mathbf{x})}}$$

1.1.5 Decision Trees

A non-parametric model that uses a tree-like structure of decisions to classify instances. It works by recursively partitioning the data space based on feature values, starting from the **root node** and branching into **internal nodes** (decisions) until it reaches a **leaf node** (the final class prediction). The best splits are chosen to maximize the homogeneity of the resulting subsets, often measured by **Information Gain** or **Gini Impurity**.

$$G(t) = 1 - \sum_{j=1}^{C} (P(j|t))^2$$

1.1.6 Random Forest

An **ensemble learning method** for classification and regression that operates by constructing a large number of Decision Trees during training. For prediction, it aggregates the results of all individual trees: taking a **majority vote** for classification or the average for regression. This technique is highly effective at **reducing overfitting** and improving predictive accuracy compared to a single Decision Tree.

$$\hat{y} = \text{mode}\{C_k(\mathbf{x})\}_{k=1}^{K}$$

These are some of examples of Supervised Machine Learning Algorithms.

Unsupervised Learning Algorithms

Unsupervised learning models work with **unlabeled data** (X) to discover hidden patterns, inherent structures, and relationships within the data

1.1.7 K-Means Clustering

A partition-based clustering algorithm that aims to group data into K pre-defined, non-overlapping clusters. It works iteratively: 1) It assigns each data point to the cluster with the nearest **mean (centroid)**. 2) It recalculates the centroids as the mean of all points assigned to that cluster. This process repeats until the centroids no longer move. The algorithm's objective is to **minimize inertia**, also known as the **within-cluster sum of squares (WCSS)**.

$$\min \sum_{k=1}^{K} \sum_{\mathbf{x} \in S_k} ||\mathbf{x} - \mu_k||^2$$

1.1.8 Hierarchical Clustering

An algorithm that builds a hierarchy of nested clusters, which is visualized as a **dendrogram** (a tree diagram). It can be **Agglomerative** (bottom-up), where each point starts as its own cluster and the closest clusters are progressively merged, or **Divisive** (top-down), where all points start in one cluster that is recursively split. The merging/splitting is determined by a **linkage criterion** (e.g., single, complete, average) that defines the distance between clusters.

$$d(A, B) = \min_{\mathbf{x} \in A, \mathbf{z} \in B} d(\mathbf{x}, \mathbf{z})$$

1.1.9 Principal Component Analysis (PCA)

A linear **dimensionality reduction** technique used to compress data while retaining the maximum amount of information (variance). It transforms the original features into a new set of orthogonal (uncorrelated) features called **principal components**. The first principal component is the direction in the data that captures the most variance, the second captures the next most, and so on.

$$\mathbf{w}_{(1)} = \underset{||\mathbf{w}||=1}{\operatorname{argmax}} \left\{ \sum_{i=1}^{N} (\mathbf{x}_i \cdot \mathbf{w})^2 \right\}$$

1.1.10 t-distributed Stochastic Neighbor Embedding (t-SNE)

A powerful non-linear **dimensionality reduction** technique used primarily for **visualizing** high-dimensional data in 2D or 3D. It models the probability distribution of neighbors in the high-dimensional space and attempts to find a low-dimensional mapping that preserves these neighbor similarities. It is particularly effective at revealing cluster structures that other methods might miss.

$$\min D_{KL}(P||Q) = \sum_{i} \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Each of these algorithms offers a unique approach to finding patterns, making predictions, and unlocking insights from data.

CHAPTER 2 – TRAINING WORK UNDERTAKEN

2.1 Project Intro

This project focuses on developing a machine learning model to classify **mushrooms** as either **edible** or **poisonous** based on their physical characteristics. The classfication of mushrooms is an important problem in the field of mycology and public health, as consuming poisonous mushrooms can lead to serious health consequences. The model developed in this project uses **Logistic Regression**, a populat supervised leaning algorithm for binary classification tasks.

Logistic Regression is a statistical method that models the probaility of a binary outcome by fitting data to a logistic curve. Despire its name suggesting regression, it is primarily used for classfication problems. The algorithm works by computing a weighted sum of input features and passing it through a sigmoid function to produce a probabily value between 0 and 1. In this project, the model classified mushrooms into two categories: edible (represented as 0) and poisonous (represented as 1).

The choice of Logistic Regression for this problem is suitable because it provides interpretable results, handles binary classification efficiently, and works well when features have a linear relationship with the log-odds of the outcome. The model is trained on a comprehensive dataset containing various mushroom attributes such as cap shape, gill color, and habitat, which serve as predictive features for determining whether a mushroom is safe to eat.

2.2 Importing Libraries and Tools

To build and evaluate the mushroom classification model, several key Python libraries were imported. Each library serves a specific and crucial function in the data science pipeline.

- **Pandas:**
  - **Source :** import pandas as pd
  - **Use :** Pandas is the cornerstone of data manipulation and analysis in Pytho. In this project, it was used to read the **mushrooms.csv** file into a structured DataFrame, which is a two-dimensional table-like data structure. It was also used for initial data inspection (e.g, using **.head()** and **.sample()**) and for creating a final comparison DataFrame to analyze the model's predictions.
- **Numpy :**
  - **Source:** import numpy as np
  - **Use:** Numpy (Numerical Python) is the fundamental package for scientific computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions. While not used

extensively for direct computation in this notebook, it is a core dependency for Pandas and Scikit-learn and is essential for handling numerical data arrays.

- **Matplotlib & Seaborn :**
  - **Source:** import matplotlib.pyplot as plt and import seaborn as sns
  - **Use:** These two libraries are used for data visulatization.
    - **Matplotlib** is a comprehensive libray for creating static, animated, and interactive visualizations. It was used as the underlying engine for plotting.
    - **Seaborn** is a higher-level interface build top of Matplotlib that provides a more attarctive and informative statistical graphics. In this project, Seaborn was used to crate *countplot* bar charts to visualize the distribution of classes and *heatmap* to display the confusion matrix, making the results easier to interpret.

- **Scikit-Learn (sklearn) :**
  - **Import Statements:**
    - *from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder, LabelEncoder.*
    - *from sklearn.compose import ColumnTransformer*
    - *from sklearn.pipeline import Pipeline*
    - *from sklearn.preprocessing import StandardScaler*
    - *from sklearn.model_selection import train_test_split*
    - *from sklearn.linear_model import LogisticRegression*
    - *from sklearn.metrics import accuracy_score, precision_score,classification_report, confusion_matrix*

  - **Source:** Scikit-learn (also known as sklearn) is imported from PyPI and is a comprehensive machine learning library built on NumPy, SciPy, and Matplotlib. The project started as a Google Summer of Code project in 2007 by David Cournapeau and has since become the most popular machine learning library in Python.

- **Purpose and Use:** Scikit-Learn provideds a wide range of tools for machine learning and statistical modelling. In this project, multiple modules from scikit-learn are used for different purposes:
    - **Preprocessing Module:** Contains tools for data transformation. The *LabelEncoder* converts categorical text labels into numerical values, *OneHotEncoder* creates binary columns for categorical features, *OrdinalEncoder* conerts categories into ordered integers, and *StandardScaler* normalizes features to have zero mean an unit variance.
    - **Compose Module:** The *ColumnTransformer* allows us to apply different preprocessing steps to different columns of the dataset, which is essential when dealing with mixed data types.
    - **Pipeline Module:** Enables the creating of a sequence of data transformation steps that can be executed together, ensuring consistency between training and testing data preprocessing.
    - **Model Selection Module:** The *train_test_split* function divides the dataset into training and testing subsets, ensuring the model can be evaluated on unseen data.
    - **Linear Model Module:** Contains the *LogisticRegression* classifier used as the primary algorithm for this binary classification task.
    - **Metrics Module:** Provides functions to evaluate model performance icnluding *accuracy_score*(overall correctness), *precision_score*(positive prediction accuracy), *classfication_report*(detailed metrics), and *confusion_matrix*(prediction breakdown).

2.3 Importing the Dataset

- Dataset Source and Description

The mushroom dataset used in this project was obtained from **Kaggle**, a popular plartform for data science and machine learning competitions. The dataset is provided by the **UCI Machine Learning Repository** and was contributed to Kaggle by **Meg Risdal.** The dataset can be accessed at: **Dataset**

This comprehensive dataset contains information about various species of mushrooms from the *Agaricus* and *Lepiota* families. The dataset was originally donated by UCI Machine Learning Repository in 1987 and has since become a classic benchmark dataset for classification tasks in machine learning education and research.

- Dataset Characteristics

  The mushroom dataset is a binary classification dataset where each sample represents a single mushroom specimen. The target variable is the class label, which indicates wheter the mushroom is edible or poisonous. The dataset contains **8,214 observations** (individual mushroom samples) with **23 attributes** describing various physical characteristics of the mushrooms.

  All features in the dataset are categorical in nature, meaning they represent discrete categories rather than continuous numerical values. Each features is encoded using single-letter abbreviations. For example, cap shapes might be represented as ‘b’ for bell, ‘c’ for conical, ‘x’ for convex, and so on.

- Dataset Features

  The dataset includes a wide range of mushroom characteristics that help distinguish between edible and poisonous species:

  - **Class:** The target variable indicating whether the mushroom is edible (e) or poisonous (p).
  - **Cap Features:** Shape, surface texture, and color of the mushroom cap
  - **Gill Features:** Attachemtn type, spacing, size, and color of the gills
  - **Stalk Features:** Shape, root type, surface texture (above and below ring), and color (above and below ring)
  - **Veil Features:** Type and color of the veil
  - **Spore Features:** Color of spore print
  - **Physical Characteristics:** Presence of bruises and odor
  - **Habitat Information :** Population type and habitat where the mushroom grows

- Loading the Dataset

  In the project, the dataset is loaded using Pandas’ *read_csv()* function :

```
df = pd.read_csv("mushrooms.csv")
```

This command reads the CSV file named 'mushrooms.csv' from the working directory and stores it in a DataFrame object named *df*. The DataFrame structure allows easy manipulation and analysis of the mushroom data, providing access to all rows and columns with labeled indices.

- Dataset Relevance

  This dataset is particularly valuable for machine learning classification tasks because it contains no missing values, has a balanced distribution of classes, and includes multiple categorical features that requires encoding techniques. The clear binay outcome (edible or poisonous) makes it an ideal dataset for demonstrating classification algorithms and evaluating their performance in a real-world context where accurate predictions have significant practical implications.

2.4 Cleaning the Dataset and Performing Exploratory Data Analysis

Data Cleaning and explorator data analysis (EDA) are critical steps in any machine learning project. These processes help us understand the structure of the data, identify potential issues, and gain insights that inform feature engineering and model selection decisions.

- **Initial Data Exploration**

  The first step in understanding the dataset involves examining its basic structure and characteristics. This project uses several Pandas functions to explore the data:

  Data Preview: The *df.head()* and *df.sample(5)* functions are used to display the first few rows and random samples of the dataset respectively. This provides a quick ovreview of the data format and the types of values present in each column.

  Dataset Dimensions: The *df.shape* attribute reveals the dimensions of the dataset, showing the total number of rows and columns.

  Data types and Structure: The *df.info()* method provides a concise summary of the DataFrame, including the number of non-null entries in each column, the data types, and memory usage. This helps identify if there are any missing values or unexpected data types.

- **Data Quality Assesment**

  A crucial aspect of data cleaning is checking for missing or null values that could affect model performance.

  Missing Value Detection : The project uses *X.isnull().sum()* and *y.isnull().sum()* to count missing values in both the feature set and the target variable. In this particulat dataset, there are no missing values, which simplifies the preprocessing pipeline and eliminates the need for imputation strategies.

  Categorical Feature Identification : The following code:

```
cat_col = df.select_dtypes(include=['object','category']).columns

print(cat_col)
```

Identifies all columns containing categorical data. Since the mushroom dataset consists entirely of categorical features, this step confirms that appropriate encoding methods will be needed before model training.

Unique Values Examination : A loop through all the categorical columns using:

```
for col in cat_col :
    print(f"\nColumn: {col}")
    print(df[col].unique())
```

displays all unique values in each feature. This helps understand the cardinality of each feature (number of distinct categories) and identify any inconsistencies or unexpected values in the data.

- **Data Splitting**

Before proceeding with visualization and preprocessing, the dataset is split into features (X) and target variable (y) by using :

```
X = df.drop(columns=['class'])
y = df['class']
```

The feature matrix X contains all 22 predictor variables, while the target vector y contains the class labels. This separation is esseintial for supervised learning where the model learns to predict y based on the patterns in X.

- **Exploratory Data Analysis with Visualizations**

The project employs both Matplotlib and Seaborn libraries to create informative visualization that reveal patterns and relationships in the data.

  - **Class Distribution Analysis**

    Visualization Type: Count Plot

    Purpose: To examine the balance between edible and poisonous classes

```
sns.countplot(x='class',data=df)
plt.title("Class Distribution : edible(0) vs Poisnous (1)")
plt.xlabel('Class')
plt.ylabel('Count')
plt.show()
```

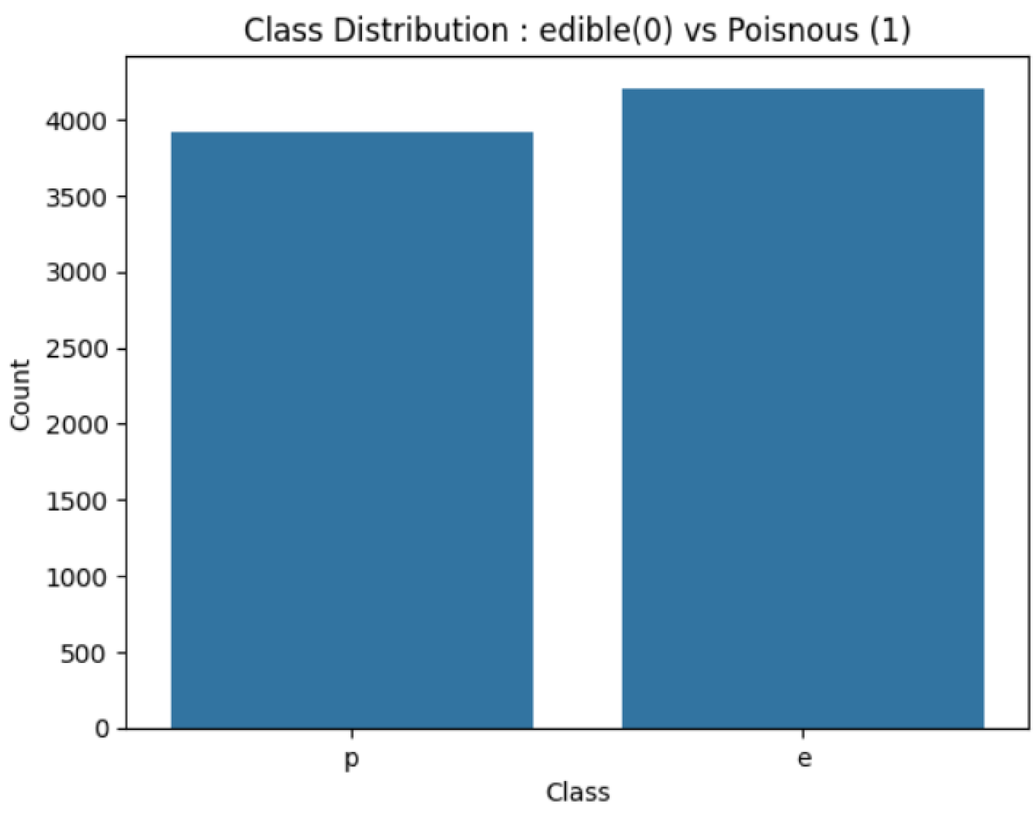


Fig 2.1 Countplot betweeen Edible and Poisonous

This visualization displays the frequency of each calss in the dataset. A balanced dataset is ideal for classification tasks, as it prevents the model from being biased toward the majority class. If there is significant class imbalance, techniques like oversampling, undersampling, or adjusting class weights might be necessary.

- **Feature Relationship Analysis**

Visualization type: Count Plot with Hue

Purpose: To explore the relationship between gill color and mushroom edibility.

```
sns.countplot(x='gill-color',hue ='class',data=df)
plt.title('Gill color vs Class')
plt.xlabel('Gill color(encoded)')
plt.ylabel('Count')
plt.legend(title='Class')
plt.show()
```

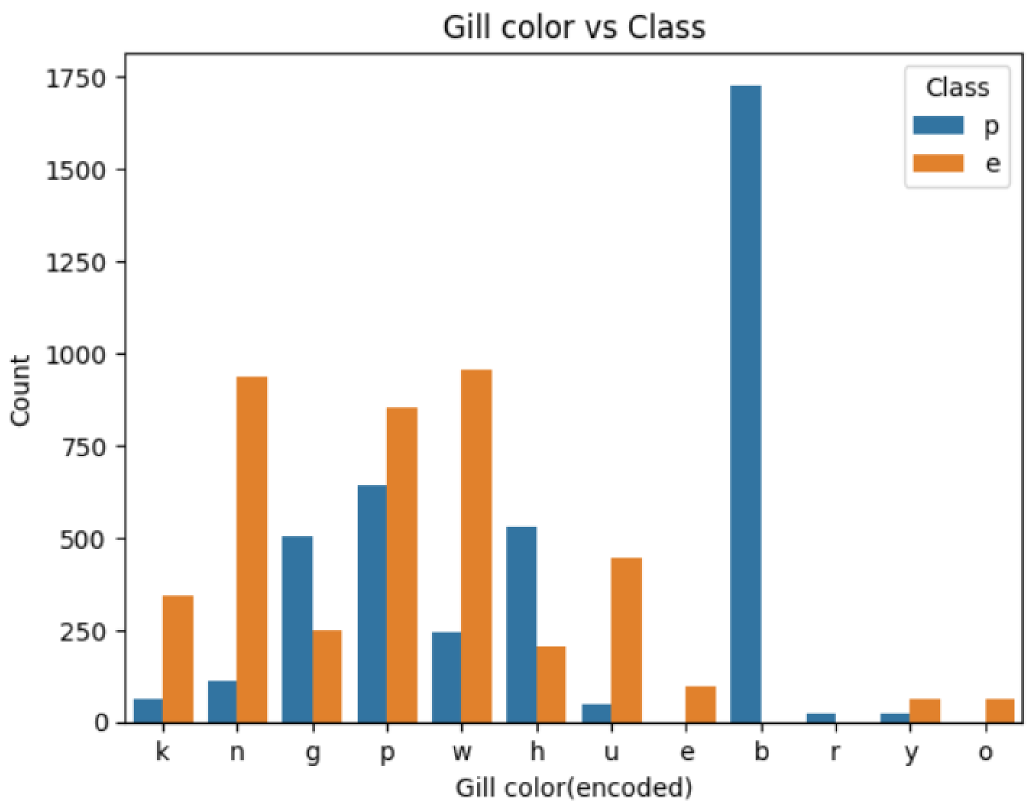


fig. 2.2 Countplot between gill colors and class

This grouped bar chart shows how different gill colors are distributed across edible and poisonous mushrooms. By using the *hue* parameter set to ‘class’, the plot creates separate bars for each class within each gill color category.

2.5 Applying the Algorithm

This section covers the core machine learning workflow, including data preprocessing, model training, and evaluation.

- **Data Preprocessing**

  Since the Logistic Regression algorithm requires numerical input, the categorical features had to be converted into a numerical format.

  Target Variable Encoding: The target variable *y* was converted from categorical labels to numerical labels using Scikit-learn’s *LabeleEncoder*

  Feature Variable Encoding: A hybrid strategy was used for the 22 feature columns.

  - **One-Hot Encoding** was applied to features with few unique categories and no intrinsic order. This method creates a new bnary column for each category.
  - **Ordinal Encoding** was applied to features with a larger number of categories or a potential order. This method converts each unique category into an integer.

  Pipeline and Scaling:

  - A *ColumnTransformer* was used to apply the correct encoding method to the specified columns.
  - This transformer was then placed into a *Pipeline,* which chained the encoding step with a *StandardScaler.* The *StandardScaler* standardizes the features by removing the mean and sclaing to unit variance. This is a crucial step for Logistic Regression, as it helps the algorithm converge faster and perform more effectively, The pipeline ensured that all transformation were applied consistently to the data.

    After processing, the original 22 features were transformed into 32 numerical features.

- **Model Training: Logistic Regression**
  - **Train-Test Split:** After completing the preprocessing stage, the cleaned dataset was divided into two distinct subsets — a **training set** and a **testing set** — using an 80:20 ratio. This means that 80% of the data was utilized for training the model, while the remaining 20% was reserved for evaluating its performance. The *stratfy* parameter was specifically employed during this process to maintain the original class distribution of edible and poisonous mushrooms across both subsets. This ensured that the proportion of each class remained consistent in the training and testing data, thereby preserving the overall balance and preventing any bias that might arise from unequal class representation. Such a step is crucial in classification tasks, as it allows the model to learn and generalize more effectively without being skewed toward a particular category.

  - **Algorithm Application :** Once the data was properly split, a **Logistic Regression** model was chosen, instantiated, and trained using the training dataset. Logistic Regression is a widely used supervised learning algorithm, particularly effective for binary classification problems like distinguishing between edible and poisonous mushrooms. The algorithm functions by finding the optimal linear relationship between the input features and the target variable, effectively determining a **decision boundary** that best separates the two classes. It estimates the probability of a mushroom belonging to each category based on its attributes and assigns the class label with the higher probability. This probabilistic approach not only allows for clear classification but also provides insight into the confidence of the model's predictions, making Logistic Regression both interpretable and reliable for this kind of predictive analysis.

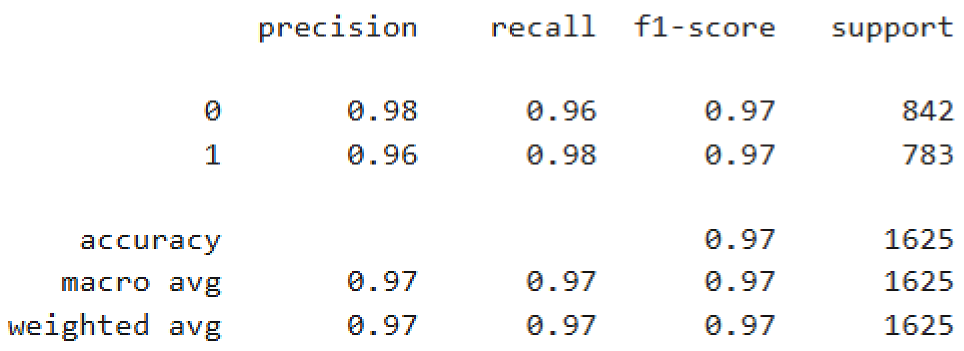
CHAPTER 3 – RESULTS AND EVALUATION

This section details the performance of the trained Logistic regression model. After training on 80% of the data, the model's predictive power was assessed using the remaining 20% (the test set) , which it had never seen before. The results demonstrate a high degreee of accuracy and reliability.

3.1 Model Performance Metrics

To measure performance, we used several standard statistical metrics. The results from the test set were as follows:

- **Accuracy:** The model achieved on an overall accuracy of **96.9%.** This means that it correctly identified whether a mushroom was edible or poisonous in nearly 97 out of every 100 cases.
- **Precision:** The precision score was approximately **95.8%.** This is a crucial metric for this problem, as it measures the model's reliability when it predicts a mushroom is poisonous. In this case, when the model flagged a mushroom as poisonous, it was correct about 96% of the time.
- **Classification Report:** The full classification report provided a detailed breakdown, showing high precision, recall, and F1-scores for both the edible and poisonous classes. This indicates that the model is not only accurate overall but is also equally skilled at identiffying both types of mushrooms, rather than being biased toward one.



```
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       842
           1       0.96      0.98      0.97       783

    accuracy                           0.97      1625
   macro avg       0.97      0.97      0.97      1625
weighted avg       0.97      0.97      0.97      1625
```

Fig. 3.1 Classification Report of the model

3.2 Graphical Visualizations

Visual aids were used to better understand and interpret the model's performance.

- **Confusion Matrix provides a clear, visual summary of the model's predictions versus** the actual labels.
    - **True Positives & True Negatives:** The large numbers on the main diagonal (808 Edible, 767 Poisonous) show the number of correct predictions.
    - **False Positives & False Negatives:** The small numbers on the off-diagnol (34 and 16) show the model's few errors. This visualization makes it immediately clear that the model's correct predictions faw outweigh its incorrect ones.

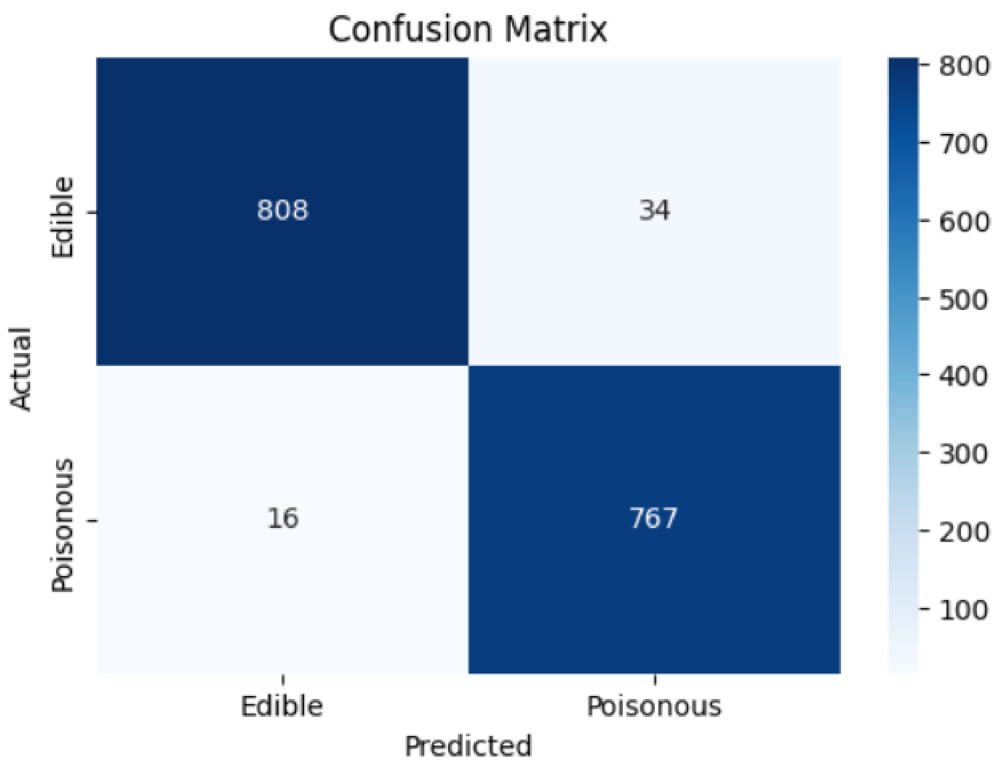


Fig. 3.2 Confusion Matrix of model performance

- **Actual vs. Predicted Counts: This bar chart directly compares the total number of** actual edible and poisonous mushrooms in the test set against the total number the model predicted for each class. The bars for "Actual" and "Predicted" are nearly identical in height for both classes, providing a simple, high-level, confirmation that the model's predictions are well aligned with the real-world data distribution.

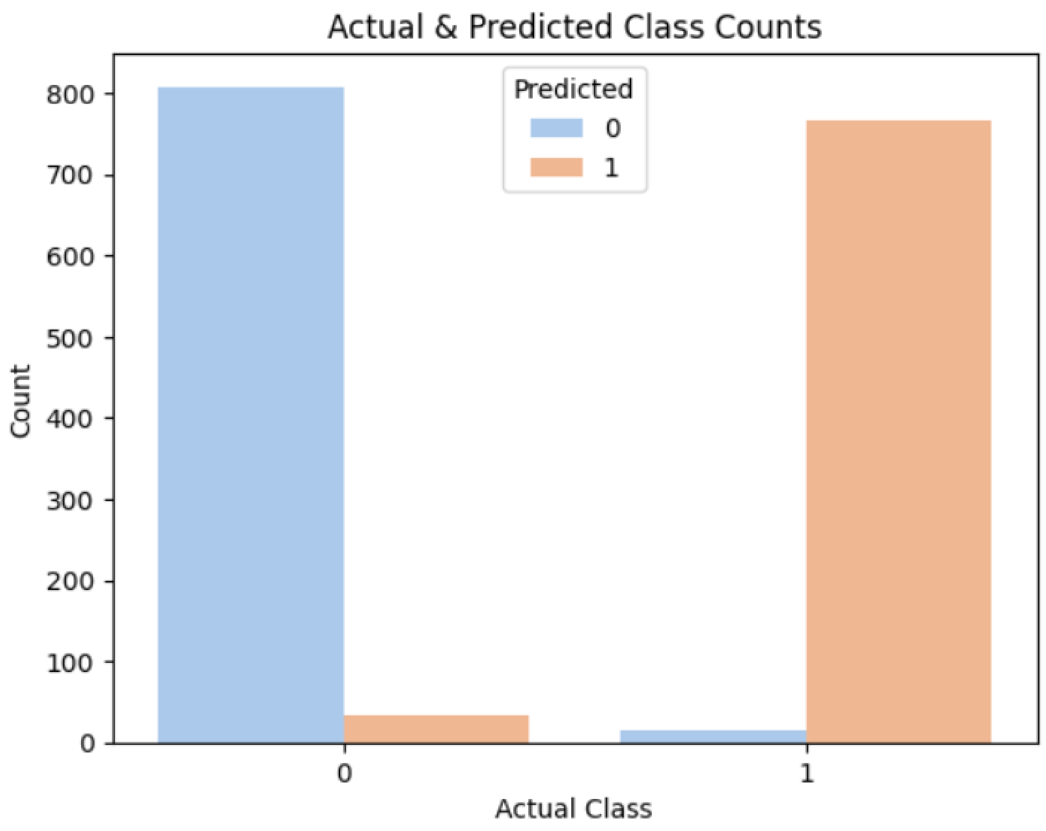


Fig 3.3 Comparison of Actual & Predicted class counts

- **Correct vs. Incorrect Predictions: This final plot breaks down the predictions for each** class into two categories : correct (True) and incorrect (False).
    - The **green bars (true)** are overwhelmingly large for both Edible and Poisonous classes, showing the high volume of correct predictions.
    - The **red bars (false)** are very small, visually highlighting the low number of erros the model made for each class. This graph effectively summarizes the model's high reliability

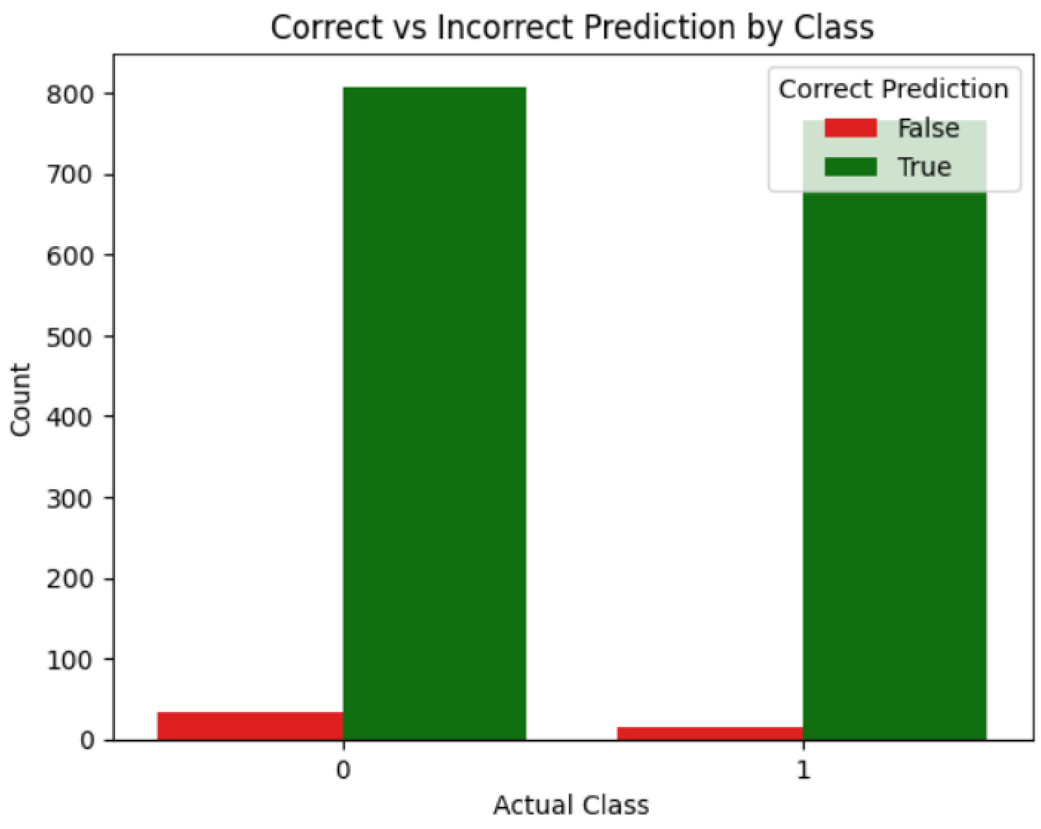


Fig 3.4 Comparing Correct vs Incorrect Predictions

CHAPTER 4 - CONCLUSIONS

4.1 Conclusion

This project successfully demonstrated the development of a highly effective machine learning model for classifying mushrooms as either **edible** or **poisonous.** By leveraging a categorical dataset from the UCI Machine Learning Repository, we performed a complete data science workflow, icnluding data cleaning, exploratory analysis, and robust preprocessing.

The **Logistic Regression** algorithm proved to be an excellent choice for this task. After training, the model achieved an outstanding accuracy **96.9%** on the unseen test data. The detailed evaluation, supported by the confusion matrix and other visualizations, confirmed the model's reliability, showing a very low number of misclassifications for both classes.

This report highlights the power of data preprocessing and standard classification algorithms in solving real-world, high-stakes problems. The model's accuracy shows it can serve as a strong predictive tool.

4.2 Future Work

While the model is highly successful, a *ConvergenceWarning* was noted during training. Future improvements could involve.

- **Hyperparameter Tuning:** Addressing the warning by increasing the *max_iter*(maximum iterations) or experimenting with a different *solver* for the Logistic Regression model.
- **Exploring Other Models:** Applying other classification algorithms, such as **Decision Trees, Random Forests,** or **Support Vector Machines(SVMs),** to see if the accuracy can be pushed even further, potentially to 100%.

4.3 References

Here are the key resources, datasets, and libraries used in this project:

1. UCI Machine Learning Repository, "Mushroom Classification Dataset," *Kaggle*. [Online]. Available: https://www.kaggle.com/datasets/uciml/mushroom-classification.
2. The Pandas Development Team, *Pandas: Python Data Analysis Library*. [Online]. Available: https://pandas.pydata.org/docs/.
3. The NumPy Developers, *NumPy: Numerical Python*. [Online]. Available: https://numpy.org/doc/.
4. J. D. Hunter, "Matplotlib: A Collaborative Visualization Library," *Matplotlib*. [Online]. Available: https://matplotlib.org/stable/users/index.html.
5. M. Waskom, *Seaborn: Statistical Data Visualization*. [Online]. Available: https://seaborn.pydata.org/.
6. F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011. [Online]. Available: https://scikit-learn.org/stable/index.html.