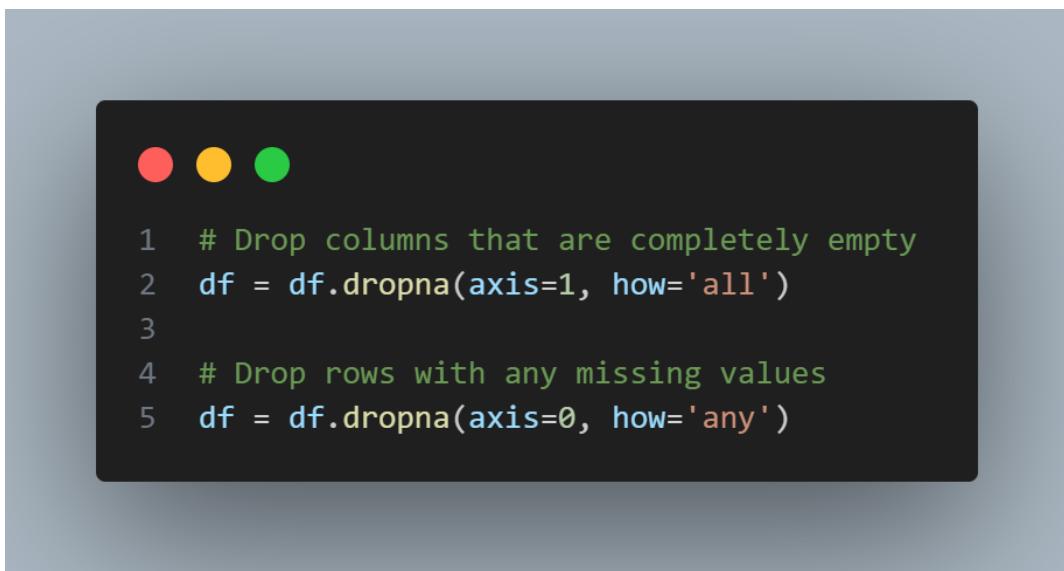


# Data Science Lab1

<b>Team nr:</b> Insert here	<b>Student 1:</b> Antero Morgado <b>IST nr:</b> 1119213
	<b>Student 2:</b> David Ferreira <b>IST nr:</b> 1107077
	<b>Student 3:</b> José Fernandes <b>IST nr:</b> 1103727
	<b>Student 4:</b> Olha Buts <b>IST nr:</b> 1116276

## CLASSIFICATION

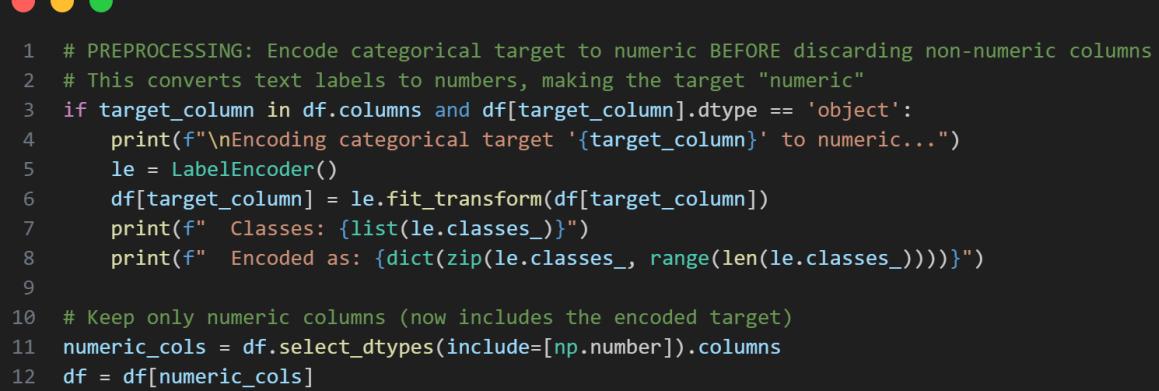
### 1 DATA CLEANING



The screenshot shows a Jupyter Notebook cell with a dark theme. At the top, there are three colored circles: red, yellow, and green. Below them is a block of Python code:

```
1 # Drop columns that are completely empty
2 df = df.dropna(axis=1, how='all')
3
4 # Drop rows with any missing values
5 df = df.dropna(axis=0, how='any')
```

Figure 1: Dropping all empty variables and all records with missing values.



```

1 # PREPROCESSING: Encode categorical target to numeric BEFORE discarding non-numeric columns
2 # This converts text labels to numbers, making the target "numeric"
3 if target_column in df.columns and df[target_column].dtype == 'object':
4     print(f"\nEncoding categorical target '{target_column}' to numeric...")
5     le = LabelEncoder()
6     df[target_column] = le.fit_transform(df[target_column])
7     print(f" Classes: {list(le.classes_)}")
8     print(f" Encoded as: {dict(zip(le.classes_, range(len(le.classes_))))}")
9
10 # Keep only numeric columns (now includes the encoded target)
11 numeric_cols = df.select_dtypes(include=[np.number]).columns
12 df = df[numeric_cols]

```

Figure 2: Discard all non-numeric data excluding the target variable.

## 2 RESULTS

### 2.1 Model Performance Summary

Table 1: Performance Metrics - Traffic Accidents Dataset

Model	Accuracy	Precision	Recall	F1-Score
Naïve Bayes	0.999	0.999	0.999	0.999
Logistic Regression	1.000	1.000	1.000	1.000
KNN	1.000	1.000	1.000	1.000
Decision Tree	1.000	1.000	1.000	1.000
Multi-layer Perceptron	1.000	1.000	1.000	1.000

Table 2: Performance Metrics - Combined Flights 2022 Dataset

<b>Model</b>	<b>Accuracy</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-Score</b>
Naïve Bayes	-	-	-	-
Logistic Regression	-	-	-	-
KNN	-	-	-	-
Decision Tree	-	-	-	-
Multi-layer Perceptron	-	-	-	-

## 2.2 Best Hyperparameters

Table 3: Optimal Hyperparameters - Traffic Accidents

<b>Model</b>	<b>Best Hyperparameters</b>
Naïve Bayes	var_smoothing: 1e-10
Logistic Regression	C: 215.44, solver: lbfgs, max_iter: 1000
KNN	n_neighbors: 5, weights: distance, metric: manhattan
Decision Tree	criterion: gini, max_depth: 5, min_samples_split: 2, min_samples_leaf: 1
Multi-layer Perceptron	hidden_layer_sizes: (50,), activation: tanh, alpha: 0.0001, learning_rate: constant

Table 4: Optimal Hyperparameters - Combined Flights 2022

<b>Model</b>	<b>Best Hyperparameters</b>
Naïve Bayes	-
Logistic Regression	-
KNN	-
Decision Tree	-
Multi-layer Perceptron	-

## 2.3 Naïve Bayes

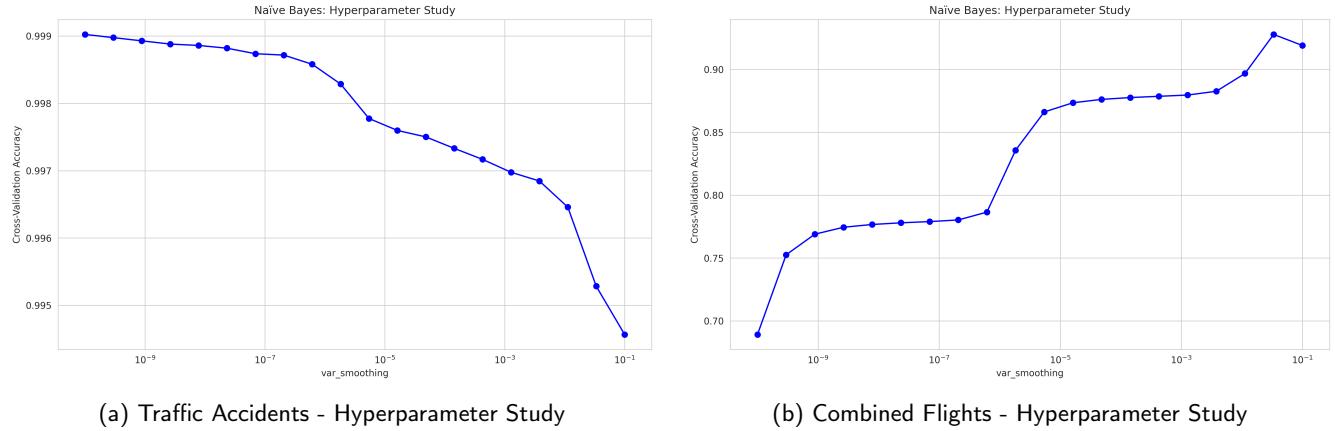


Figure 3: Naïve Bayes: Hyperparameter Tuning Results

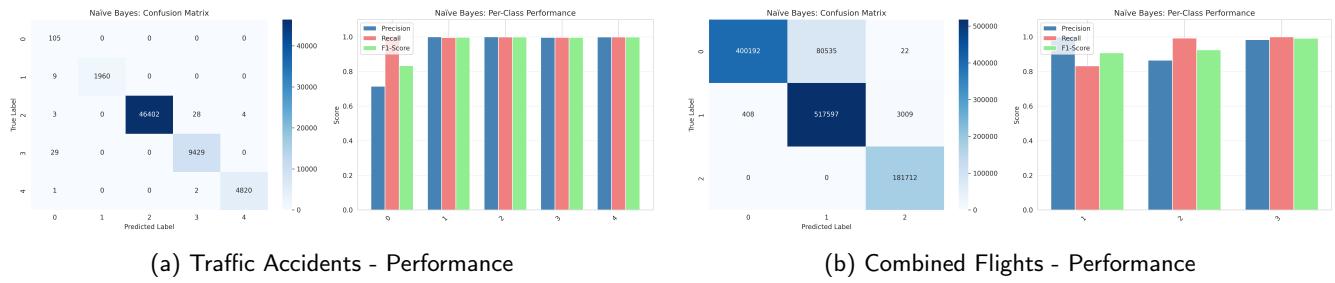


Figure 4: Naïve Bayes: Model Performance

## 2.4 Logistic Regression

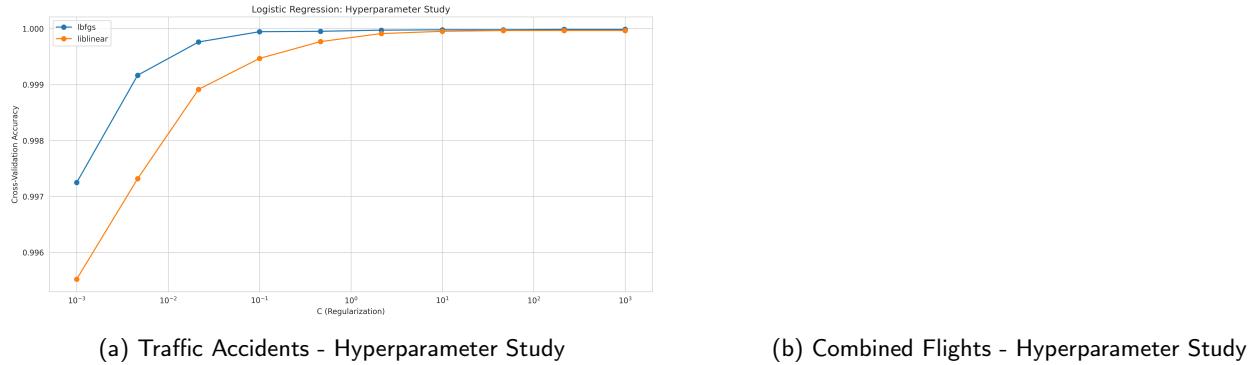


Figure 5: Logistic Regression: Hyperparameter Tuning Results

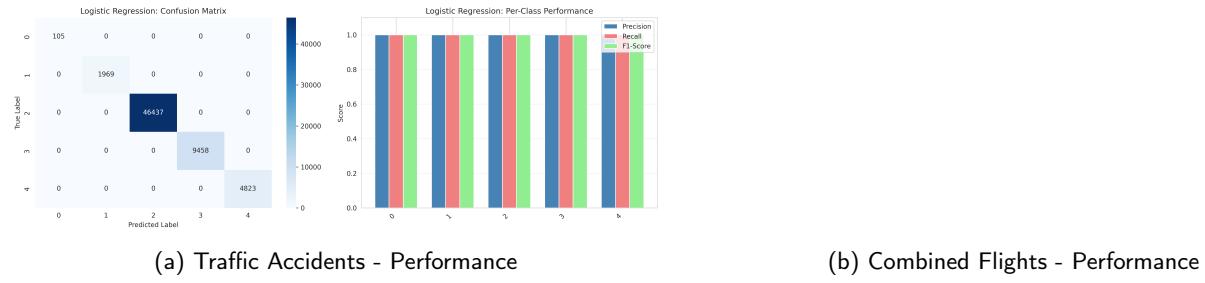


Figure 6: Logistic Regression: Model Performance

## 2.5 K-Nearest Neighbors (KNN)

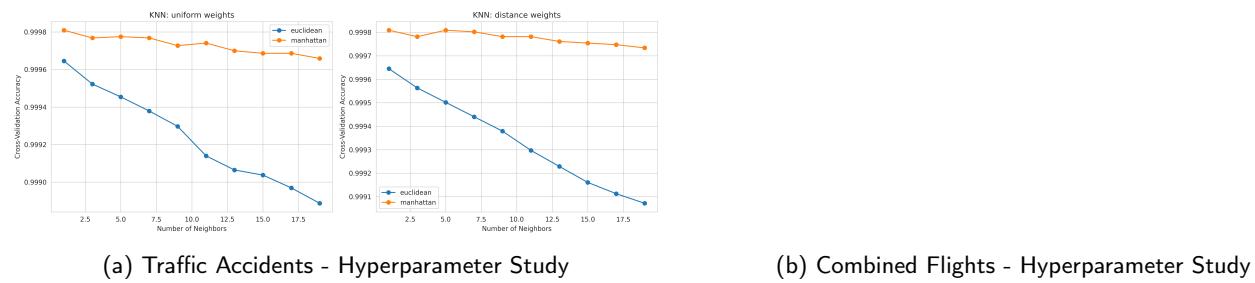


Figure 7: KNN: Hyperparameter Tuning Results

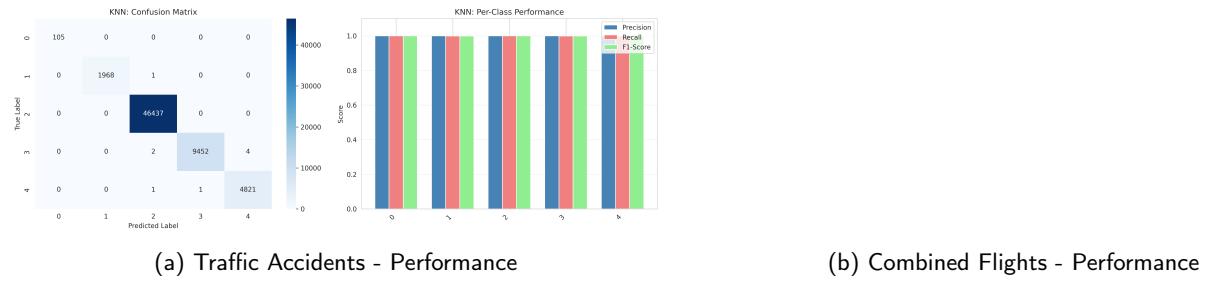


Figure 8: KNN: Model Performance

## 2.6 Decision Tree

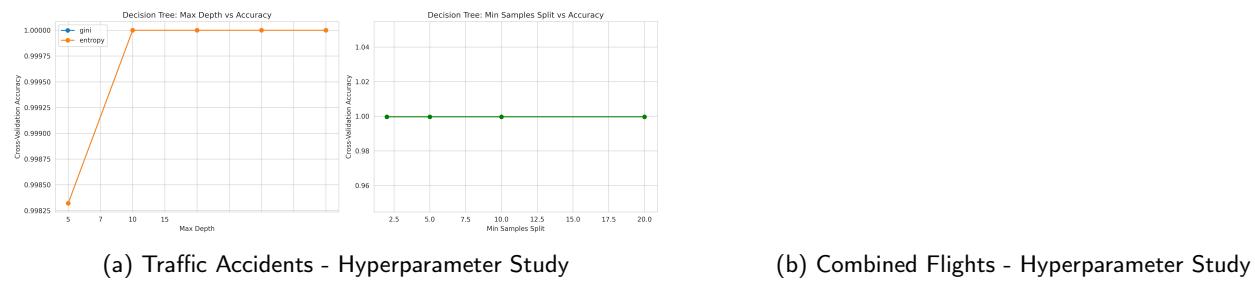


Figure 9: Decision Tree: Hyperparameter Tuning Results



Figure 10: Decision Tree: Model Performance

## 2.7 Multi-layer Perceptron

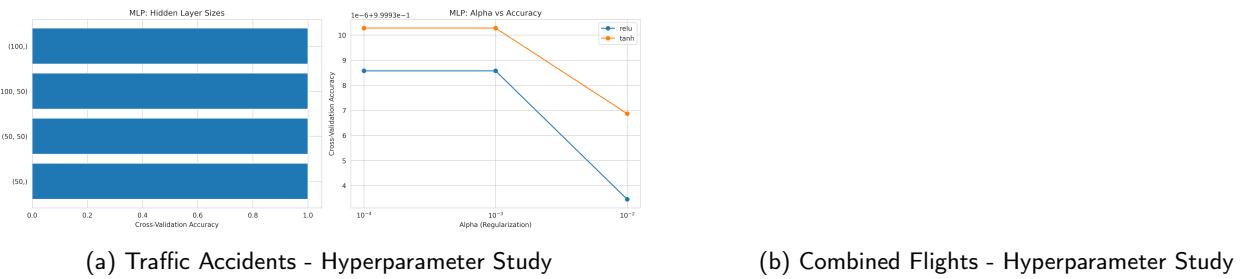


Figure 11: Multi-layer Perceptron: Hyperparameter Tuning Results



Figure 12: Multi-layer Perceptron: Model Performance

## 2.8 Overall Model Comparison

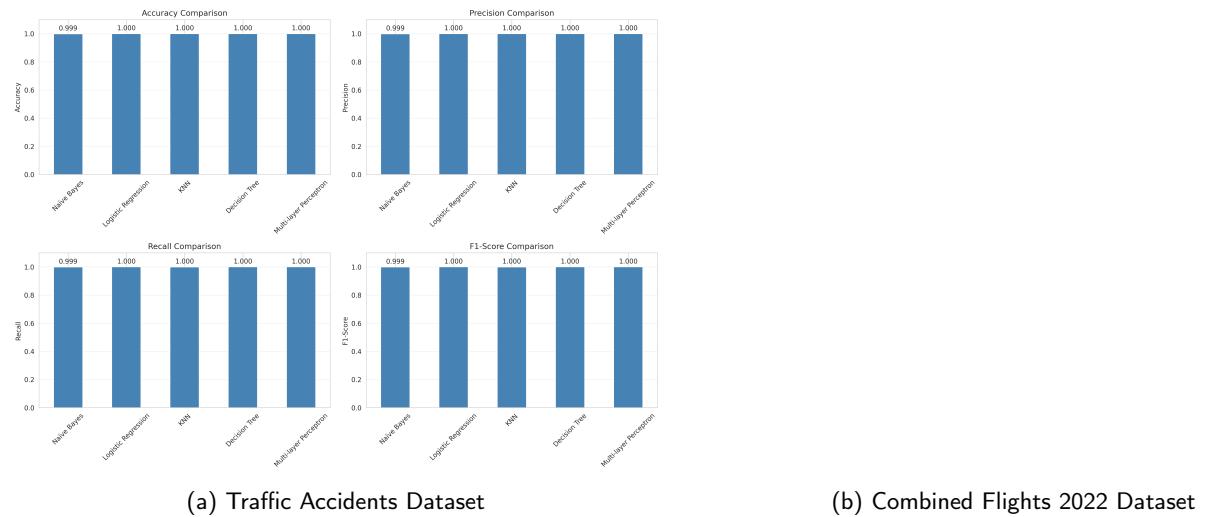


Figure 13: Overall Performance Comparison Across All Models