**Predicting Cosmic Radiation Impact On Spacecraft Electronics Using AI**

**30% code**

import pandas as pd

import os

from skimage.transform import resize

from skimage.io import imread

import numpy as np

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

import seaborn as sns

from skimage import io, transform

from sklearn import preprocessing

import numpy as np

import joblib

import cv2

import warnings

warnings.filterwarnings('ignore')

from imblearn.over\_sampling import SMOTE

from sklearn.metrics import accuracy\_score,confusion\_matrix,classification\_report

df = pd.read\_csv('cosmic\_radiation\_impact\_equal.csv')

df

df['Impact\_Severity'].unique()

df.isnull().sum()

df.duplicated().sum()

df.info()

x = df.drop(['Impact\_Severity'],axis=1)

x

y = df['Impact\_Severity']

y

sns.countplot(x=y)

from sklearn.preprocessing import LabelEncoder

label\_encoder = LabelEncoder()

df['Impact\_Severity'] = label\_encoder.fit\_transform(df['Impact\_Severity'])

df

df['Impact\_Severity'].unique()

labels = ['Normal','Mild Impact','Severe Impact','Moderate Impaact']

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.20,random\_state=77)

**EXISTING SYSTEM**

Traditionally, the impact of cosmic radiation on spacecraft electronics has been studied using physics-based models and radiation testing in simulated environments. Engineers rely on Monte Carlo simulations, statistical analysis, and empirical space weather models to estimate radiation effects. Shielding materials like aluminum or polyethylene are used to protect critical electronic components. Space agencies also conduct ground-based testing using particle accelerators to expose electronics to radiation levels similar to those in space. Additionally, radiation-hardened (rad-hard) components are developed to withstand high-energy cosmic rays, but these components are expensive and may lag behind commercial advancements in semiconductor technology.

**Limitations**

1. **High Computational Cost** – Monte Carlo simulations and physics-based models require extensive computation, making them slow and resource-intensive.
2. **Limited Real-Time Adaptability** – Traditional methods cannot dynamically adapt to changing space weather conditions in real-time.
3. **Expensive Radiation-Hardened Components** – Specialized rad-hard electronics are costly and often have lower performance compared to commercial off-the-shelf (COTS) alternatives.
4. **Static Shielding Strategies** – Fixed shielding designs do not optimize protection based on real-time radiation exposure, leading to **overdesign (excess weight)** or **underdesign (component failures)**.
5. **Lack of Predictive Precision** – Empirical models may not accurately predict rare but catastrophic radiation-induced failures such as **Single Event Upsets (SEUs)** or **Total Ionizing Dose (TID) effects**.
6. **Limited Data Utilization** – Traditional approaches do not fully leverage large datasets from past missions and radiation monitoring satellites to enhance prediction accuracy.

**PROPOSED METHODOLOGY**

**Logistic Regression Classifier**

Logistic Regression is a supervised machine learning algorithm used for binary classification problems, meaning it predicts one of two possible outcomes (e.g., spam vs. not spam, disease vs. no disease, fire vs. no fire). Despite its name, logistic regression is a classification algorithm, not a regression algorithm. It works by estimating the probability that a given input belongs to a certain class.

Logistic regression models the relationship between the **input features (X)** and the **output (Y)** using the **logistic (sigmoid) function**:

P(Y=1∣X)=11+e−(β0+β1X1+β2X2+...+βnXn)P(Y=1 | X) = \frac{1}{1 + e^{-(\beta\_0 + \beta\_1X\_1 + \beta\_2X\_2 + ... + \beta\_nX\_n)}}P(Y=1∣X)=1+e−(β0​+β1​X1​+β2​X2​+...+βn​Xn​)1​

where:

* P(Y=1∣X)P(Y=1 | X)P(Y=1∣X) is the probability that the output is **1** (positive class).
* β0\beta\_0β0​ is the **intercept** (bias).
* β1,β2,...,βn\beta\_1, \beta\_2, ..., \beta\_nβ1​,β2​,...,βn​ are the **coefficients (weights)** associated with input features X1,X2,...,XnX\_1, X\_2, ..., X\_nX1​,X2​,...,Xn​.
* eee is the Euler’s number (~2.718), used in exponentiation.

The **sigmoid function** ensures the output is between **0 and 1**, making it interpretable as a probability.

**How Logistic Regression Works**

1. **Step 1: Data Collection & Preprocessing**
   * Gather labeled data (features & target variable).
   * Handle missing values, normalize/scale numerical features, and encode categorical variables if needed.
2. **Step 2: Model Training**
   * The algorithm learns the optimal **weights (β\betaβ)** by minimizing the **log loss (binary cross-entropy loss)** using **Gradient Descent**.
3. **Step 3: Prediction**
   * After training, the model uses the learned weights to compute the probability of class 1 using the **sigmoid function**.
   * If P(Y=1∣X)>0.5P(Y=1 | X) > 0.5P(Y=1∣X)>0.5, predict **class 1**, else predict **class 0**.
4. **Step 4: Model Evaluation**
   * The model’s performance is assessed using metrics like **Accuracy, Precision, Recall, F1-score, and ROC-AUC score**.

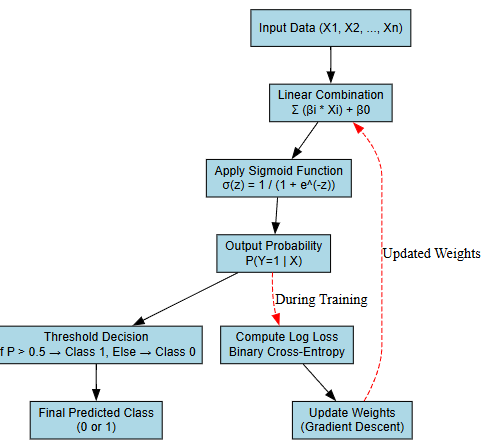


Fig 1 Block diagram flow of Logistic regression classifier

**SVM Classifier**

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It is widely used for binary classification problems and works by finding the optimal hyperplane that best separates different classes in a high-dimensional space.

**How SVM Works?**

SVM aims to find a decision boundary (hyperplane) that maximizes the margin between two classes. The key concepts behind SVM are:

**1. Finding the Optimal Hyperplane**

* A hyperplane is a decision boundary that separates different classes in an N-dimensional feature space.
* The best hyperplane is the one that maximizes the margin, which is the distance between the hyperplane and the nearest data points from both classes (called support vectors).

**2. Support Vectors**

* The data points that lie closest to the hyperplane and influence its position are called support vectors.
* Only these support vectors determine the **margin and classification**, making SVM robust to outliers.

**3. Margin Maximization**

* A **wider margin** improves generalization and reduces overfitting.
* SVM ensures that the hyperplane is positioned to **maximize the margin** between classes.

**4. Handling Non-Linearly Separable Data (Kernel Trick)**

* If data is **not linearly separable**, SVM uses the **kernel trick** to map data into a **higher-dimensional space** where it becomes linearly separable.
* Common kernel functions:
  + **Linear Kernel**: K(xi,xj)=xi⋅xjK(x\_i, x\_j) = x\_i \cdot x\_jK(xi​,xj​)=xi​⋅xj​
  + **Polynomial Kernel**: K(xi,xj)=(xi⋅xj+c)dK(x\_i, x\_j) = (x\_i \cdot x\_j + c)^dK(xi​,xj​)=(xi​⋅xj​+c)d
  + **Radial Basis Function (RBF) Kernel**: K(xi,xj)=e−γ∣∣xi−xj∣∣2K(x\_i, x\_j) = e^{-\gamma ||x\_i - x\_j||^2}K(xi​,xj​)=e−γ∣∣xi​−xj​∣∣2
  + **Sigmoid Kernel**: K(xi,xj)=tanh⁡(αxi⋅xj+c)K(x\_i, x\_j) = \tanh(\alpha x\_i \cdot x\_j + c)K(xi​,xj​)=tanh(αxi​⋅xj​+c)

**5. Mathematical Formulation**

The SVM optimization problem is:

min⁡w,b12∣∣w∣∣2\min\_{w, b} \frac{1}{2} ||w||^2w,bmin​21​∣∣w∣∣2

subject to:

yi(w⋅xi+b)≥1,∀iy\_i (w \cdot x\_i + b) \geq 1, \quad \forall iyi​(w⋅xi​+b)≥1,∀i

where:

* www = Weight vector (defining the hyperplane).
* bbb = Bias term.
* yiy\_iyi​ = Class labels (+1 or -1).

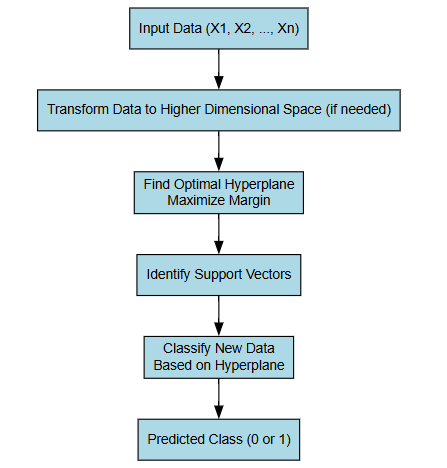


Fig 2 Block diagram flow of SVM classifier

**Block diagram**

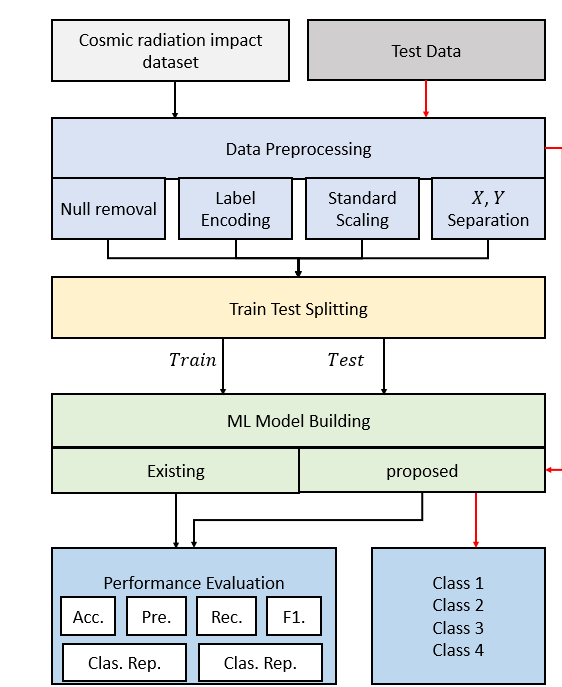
****

Fig 3 Block diagram

**Expected output**

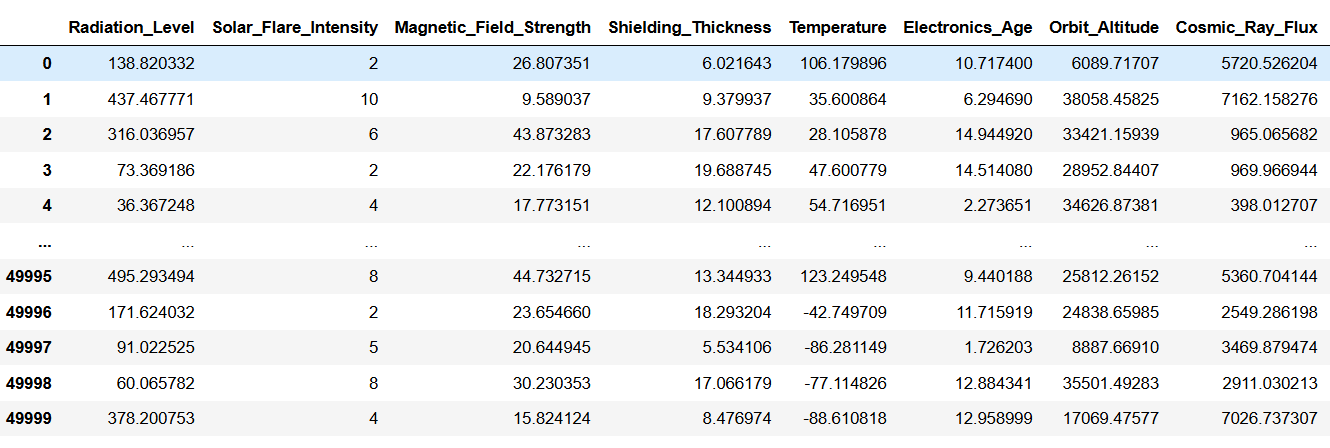
****

Fig 4 Upload dataset

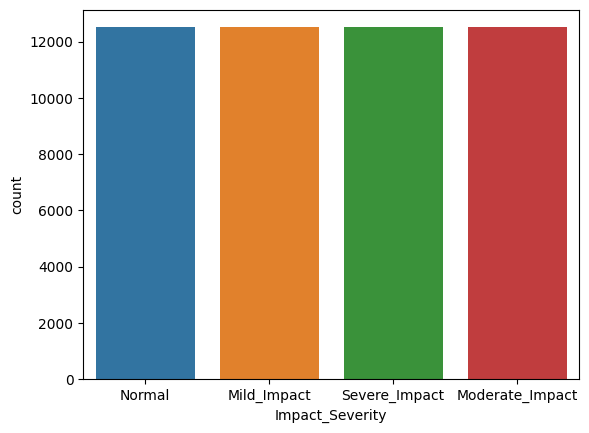


Fig 5 Countplot

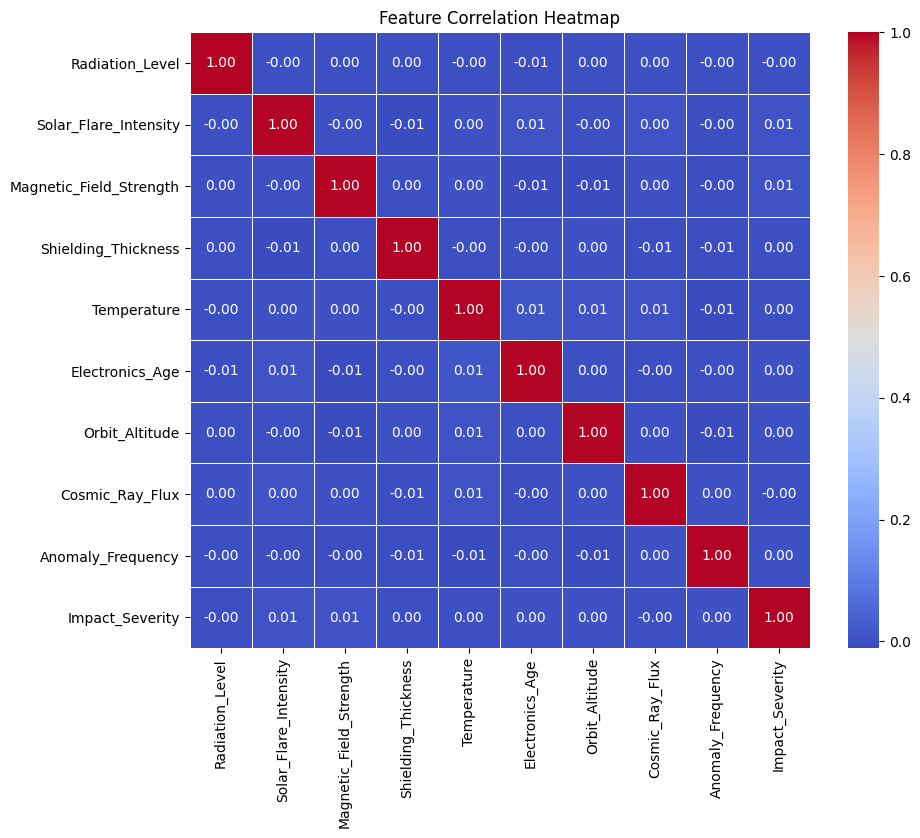


Fig 6 Correlation Heatmap