# **Project Report**

**Name:** [Chinnareddy.K]  
 **Course:** [Machine Learning and fundamentals / U24CST362]  
 **Registration Number:** [TU6243202111013]

### **Project Title:**

[Breast Cancer Diagnosis]

## **1. Project Description**

Provide a concise overview of the problem. Explain the real-world context, motivation, and the goal of the project. Mention what the project aims to achieve (e.g., prediction, classification, recommendation, clustering, etc.).

## **2. Learning Objectives**

* **Objective 1:** [State objective clearly]
* **Objective 2:** [State objective clearly]
* **Objective 3:** [State objective clearly]

## **3. Timeline**

* **Start Date:** [e.g., Sept 7, 2025]
* **Submission date:** [e.g., Sept 14, 2025]

## **4. Algorithm Used**

* **Algorithm Name:** [e.g., Logistic Regression, Random Forest, Neural Network, etc.]
* **Explanation:** Briefly describe how the algorithm works, why it was chosen, and its advantages in solving the given problem.

## **5. Tools & Libraries**

* **Programming Language:** Python
* **Libraries Used:**
  + Pandas
  + NumPy
  + Scikit-learn
  + Matplotlib / Seaborn
  + TensorFlow / PyTorch (if used)
  + [Any other relevant tool]

## **6. Dataset Description**

* **Source:** [e.g., Kaggle, UCI ML Repository, Custom Collected Data]
* **Size:** [Number of rows and features]
* **Target Variable:** [If supervised learning]
* **Description of Features:** Provide a short description of key features.

## **7. Methodology**

* **Data Preprocessing:** Handling missing values, encoding, scaling, feature selection.
* **Model Training:** Split dataset (train/test), chosen model, training procedure.
* **Evaluation:** Metrics used (Accuracy, RMSE, AUC, F1-score, etc.).
* **Hyperparameter Tuning:** If performed, mention GridSearchCV/RandomSearch/Optuna.

## **8. Results**

* **Performance Metrics:** Provide table/values of evaluation metrics.
* **Visualizations:** Include confusion matrix, ROC curve, error distribution plots, feature importance, etc.
* **Insights:** Interpret results—what do they mean in the project’s context?

Attach screenshot of outputs and short description

## **9. Questions Answered**

# Step 0: Install and import necessary libraries

!pip install pandas numpy matplotlib seaborn scikit-learn imbalanced-learn --quiet

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score, confusion\_matrix, roc\_curve

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression

from imblearn.over\_sampling import SMOTE

Step 1: Load Dataset

Assuming you have a CSV tumor\_data.csv with features like mean\_radius, mean\_texture, mean\_smoothness and target diagnosis (M=Malignant, B=Benign).

python

Copy code

# Load dataset

df = pd.read\_csv('tumor\_data.csv')

df.head()

Step 2: Explore the Dataset

python

Copy code

print(df.shape)

print(df.info())

print(df.describe())

# Check class distribution

print(df['diagnosis'].value\_counts())

# Encode target

df['diagnosis'] = df['diagnosis'].map({'M':1, 'B':0})

Step 3: Handle Missing Values

python

Copy code

# Check missing values

print(df.isnull().sum())

# Fill missing numeric values with mean (if any)

df.fillna(df.mean(), inplace=True)

Step 4: Feature Correlation

python

Copy code

# Correlation heatmap

plt.figure(figsize=(12,10))

sns.heatmap(df.corr(), annot=False, cmap='coolwarm')

plt.title("Correlation Matrix")

plt.show()

# Most predictive features

feature\_corr = df.corr()['diagnosis'].sort\_values(ascending=False)

print(feature\_corr)

Step 5: Define Features and Target

python

Copy code

X = df.drop(['diagnosis', 'id'], axis=1, errors='ignore') # Drop non-feature columns

y = df['diagnosis']

Step 6: Handle Imbalanced Data

python

Copy code

# Use SMOTE for balancing

sm = SMOTE(random\_state=42)

X\_res, y\_res = sm.fit\_resample(X, y)

print(pd.Series(y\_res).value\_counts())

Step 7: Split Data

python

Copy code

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_res, y\_res, test\_size=0.2, random\_state=42)

Step 8: Normalize Features

python

Copy code

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

Step 9: Train KNN Classifier

python

Copy code

# Choose K using cross-validation

k\_values = range(1, 21)

cv\_scores = []

for k in k\_values:

knn = KNeighborsClassifier(n\_neighbors=k)

scores = cross\_val\_score(knn, X\_train\_scaled, y\_train, cv=5, scoring='accuracy')

cv\_scores.append(scores.mean())

optimal\_k = k\_values[cv\_scores.index(max(cv\_scores))]

print(f"Optimal K: {optimal\_k}")

# Train KNN with optimal K

knn = KNeighborsClassifier(n\_neighbors=optimal\_k)

knn.fit(X\_train\_scaled, y\_train)

y\_pred\_knn = knn.predict(X\_test\_scaled)

Step 10: Train Logistic Regression

python

Copy code

lr = LogisticRegression(max\_iter=1000)

lr.fit(X\_train\_scaled, y\_train)

y\_pred\_lr = lr.predict(X\_test\_scaled)

y\_prob\_lr = lr.predict\_proba(X\_test\_scaled)[:,1]

Step 11: Model Evaluation

python

Copy code

def evaluate\_model(y\_test, y\_pred, y\_prob=None):

acc = accuracy\_score(y\_test, y\_pred)

prec = precision\_score(y\_test, y\_pred)

rec = recall\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

print(f"Accuracy: {acc:.3f}, Precision: {prec:.3f}, Recall: {rec:.3f}, F1: {f1:.3f}")

if y\_prob is not None:

roc\_auc = roc\_auc\_score(y\_test, y\_prob)

print(f"ROC-AUC: {roc\_auc:.3f}")

print("KNN Evaluation:")

evaluate\_model(y\_test, y\_pred\_knn)

print("\nLogistic Regression Evaluation:")

evaluate\_model(y\_test, y\_pred\_lr, y\_prob\_lr)

Step 12: Confusion Matrix and ROC Curve

python

Copy code

# Confusion matrix for Logistic Regression

cm = confusion\_matrix(y\_test, y\_pred\_lr)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

# ROC Curve

fpr, tpr, thresholds = roc\_curve(y\_test, y\_prob\_lr)

plt.plot(fpr, tpr, color='blue')

plt.plot([0,1],[0,1],'r--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.show()

Step 13: Visualize Decision Boundaries (for 2 features)

python

Copy code

from matplotlib.colors import ListedColormap

# Pick two features for visualization

X\_vis = X\_train\_scaled[:, :2]

y\_vis = y\_train

knn\_vis = KNeighborsClassifier(n\_neighbors=optimal\_k)

knn\_vis.fit(X\_vis, y\_vis)

# Meshgrid

x\_min, x\_max = X\_vis[:, 0].min() - 1, X\_vis[:, 0].max() + 1

y\_min, y\_max = X\_vis[:, 1].min() - 1, X\_vis[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, 0.1),

np.arange(y\_min, y\_max, 0.1))

Z = knn\_vis.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.4, cmap=ListedColormap(('green','red')))

plt.scatter(X\_vis[:,0], X\_vis[:,1], c=y\_vis, s=20, edgecolor='k')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.title('Decision Boundary (KNN)')

plt.show()

## **10. Challenges & Improvements**

* **Challenges:** Mention limitations faced during data collection, preprocessing, model fitting, or evaluation.
* **Future Improvements:** Suggest improvements such as using larger datasets, advanced algorithms, ensemble models, feature engineering, or deep learning.

## **11. References**

* **Dataset Links:** [Insert link]
* **Research Papers / Documentation:** [Insert APA/IEEE references]

## **12. GitHub Link**

[https://github.com/chinnareddy7/ml-project-giagnosis/upload]