# Classification of Breast Cancer Patients using Somatic Mutation and Machine Learning Approaches

**Overview**

This project applies various machine learning models to classify breast cancer patients based on somatic mutation profiles. The dataset (data.csv) has been limited to 358 patients for consistency with related research studies. It includes binary somatic mutation indicators, and the goal is to predict patient survival (vital.status).

## Dataset

* **Total Samples**: 358 patients
* **Target Variable**: vital.status (0 = survived, 1 = died)
* **Feature Group Used**: mu\_ = Somatic mutation (binary: 1/0)

## Tools and Libraries

* **Data Handling**: pandas, numpy
* **ML Models**: scikit-learn
* **Visualization**: matplotlib, seaborn
* **Feature Selection**: SelectKBest (f\_classif)

## Data Preprocessing

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.metrics import accuracy\_score  
from sklearn.feature\_selection import SelectKBest, f\_classif  
  
# Load data and limit to 358 samples  
data = pd.read\_csv('data.csv').head(358)  
  
# Select somatic mutation features only  
X = data[[col for col in data.columns if col.startswith('mu\_')]]  
y = data['vital.status']  
  
# Feature selection  
selector = SelectKBest(score\_func=f\_classif, k=50)  
X\_selected = selector.fit\_transform(X, y)  
selected\_features = X.columns[selector.get\_support()]  
  
# Train-test split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=42)  
  
# Scaling  
scaler = StandardScaler()  
X\_train = scaler.fit\_transform(X\_train)  
X\_test = scaler.transform(X\_test)

**✅ At the end of preprocessing:**

* The data has been limited to 358 patients.
* Only somatic mutation (mu\_) features have been retained.
* The top 50 most relevant features were selected.
* Data is now clean, numeric, and scaled — ready for modeling.

## Machine Learning Models

#### 1. K-Nearest Neighbors (KNN)

from sklearn.neighbors import KNeighborsClassifier  
knn = KNeighborsClassifier(n\_neighbors=5)  
knn.fit(X\_train, y\_train)  
knn\_acc = accuracy\_score(y\_test, knn.predict(X\_test))  
print(f"KNN Accuracy : {knn\_acc:2%}")

KNN Accuracy: 87.23%

#### 2. Random Forest

from sklearn.ensemble import RandomForestClassifier  
rf = RandomForestClassifier(n\_estimators=200, max\_depth=10, random\_state=42)  
rf.fit(X\_train, y\_train)  
rf\_acc = accuracy\_score(y\_test, rf.predict(X\_test))  
print(f"Random Forest Accuracy: {rf\_acc:.2%}")

Random Forest Accuracy: 89.36%

#### 3. Decision Tree

from sklearn.tree import DecisionTreeClassifier  
dt = DecisionTreeClassifier(max\_depth=5, random\_state=42)  
dt.fit(X\_train, y\_train)  
dt\_acc = accuracy\_score(y\_test, dt.predict(X\_test))  
print(f"Decision Tree Accuracy: {dt\_acc:.2%}")

Decision Tree Accuracy: 87.94%

#### 4. Logistic Regression

from sklearn.linear\_model import LogisticRegression  
lr = LogisticRegression(max\_iter=1000)  
lr.fit(X\_train, y\_train)  
lr\_acc = accuracy\_score(y\_test, lr.predict(X\_test))  
print(f"Logistic Regression Accuracy: {lr\_acc:.2%}")

Logistic Regression Accuracy: 87.23%

#### 5. Artificial Neural Network (ANN)

from sklearn.neural\_network import MLPClassifier  
ann = MLPClassifier(hidden\_layer\_sizes=(100, 50), max\_iter=2000, random\_state=42)  
ann.fit(X\_train, y\_train)  
ann\_acc = accuracy\_score(y\_test, ann.predict(X\_test))  
print(f"MLP Neural Network Accuracy: {ann\_acc:.2%}")

MLP Neural Network Accuracy: 88.65%

#### 6. Naive Bayes

from sklearn.naive\_bayes import GaussianNB  
nb = GaussianNB()  
nb\_acc = accuracy\_score(y\_test, nb.predict(X\_test))  
print(f"Naive Bayes Accuracy: {nb\_acc:.2%}")

Naive Bayes Accuracy: 46.81%

#### 7. Extra Trees Classifier

from sklearn.ensemble import ExtraTreesClassifier  
et = ExtraTreesClassifier(n\_estimators=200, random\_state=42)  
et.fit(X\_train, y\_train)  
et\_acc = accuracy\_score(y\_test, et.predict(X\_test))  
print(f"Extra Trees Accuracy: {et\_acc:.2%}")

Extra Trees Accuracy: 86.52%

#### 8. Gradient Boosting Classifier

from sklearn.ensemble import GradientBoostingClassifier  
gb = GradientBoostingClassifier(random\_state=42)  
gb.fit(X\_train, y\_train)  
gb\_acc = accuracy\_score(y\_test, gb.predict(X\_test))  
print(f"Gradient Boosting Accuracy: {gb\_acc:.2%}")

Gradient Boosting Accuracy: 86.52%

#### 9. AdaBoost Classifier

from sklearn.ensemble import AdaBoostClassifier  
ada = AdaBoostClassifier(random\_state=42)  
ada.fit(X\_train, y\_train)  
ada\_acc = accuracy\_score(y\_test, ada.predict(X\_test))  
print(f"AdaBoost Accuracy: {ada\_acc:.2%}")

AdaBoost Accuracy: 88.65%

#### 10. XGBoost Classifier

from xgboost import XGBClassifier  
xgb = XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss', random\_state=42)  
xgb.fit(X\_train, y\_train)  
xgb\_acc = accuracy\_score(y\_test, xgb.predict(X\_test))  
print(f"XGBoost Accuracy: {xgb\_acc:.2%}")

XGBoost Accuracy: 87.23%

#### 11. LightGBM Classifier

from lightgbm import LGBMClassifier  
lgb = LGBMClassifier(random\_state=42)  
lgb.fit(X\_train, y\_train)  
lgb\_acc = accuracy\_score(y\_test, lgb.predict(X\_test))  
print(f"LightGBM Accuracy: {lgb\_acc:.2%}")

LightGBM Accuracy: 88.65%

#### 12. Ridge Classifier

from sklearn.linear\_model import RidgeClassifier  
ridge = RidgeClassifier()  
ridge.fit(X\_train, y\_train)  
ridge\_acc = accuracy\_score(y\_test, ridge.predict(X\_test))  
print(f"Ridge Classifier Accuracy: {ridge\_acc:.2%}")

Ridge Classifier Accuracy: 87.94%

#### 13. Lasso Logistic Regression

from sklearn.linear\_model import LogisticRegression  
lasso = LogisticRegression(penalty='l1', solver='liblinear', max\_iter=1000)  
lasso.fit(X\_train, y\_train)  
lasso\_acc = accuracy\_score(y\_test, lasso.predict(X\_test))  
print(f"Lasso Regression Accuracy: {lasso\_acc:.2%}")

Lasso Regression Accuracy: 87.94%

#### 14. Stochastic Gradient Descent (SGD)

from sklearn.linear\_model import SGDClassifier  
sgd = SGDClassifier(max\_iter=1000, tol=1e-3, random\_state=42)  
sgd.fit(X\_train, y\_train)  
sgd\_acc = accuracy\_score(y\_test, sgd.predict(X\_test))  
print(f"SGD Classifier Accuracy: {sgd\_acc:.2%}")

SGD Classifier Accuracy: 87.23%

#### 15. Linear Discriminant Analysis (LDA)

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis  
lda = LinearDiscriminantAnalysis()  
lda.fit(X\_train, y\_train)  
lda\_acc = accuracy\_score(y\_test, lda.predict(X\_test))  
print(f"Linear Discriminant Analysis Accuracy: {lda\_acc:.2%}")

Linear Discriminant Analysis Accuracy: 85.11%

#### 16. Quadratic Discriminant Analysis (QDA)

from sklearn.discriminant\_analysis import QuadraticDiscriminantAnalysis  
qda = QuadraticDiscriminantAnalysis()  
qda.fit(X\_train, y\_train)  
qda\_acc = accuracy\_score(y\_test, qda.predict(X\_test))  
print(f"Quadratic Discriminant Analysis Accuracy: {qda\_acc:.2%}")

Quadratic Discriminant Analysis Accuracy: 46.81%

#### 17. Passive Aggressive Classifier

from sklearn.linear\_model import PassiveAggressiveClassifier  
pa = PassiveAggressiveClassifier(random\_state=42)  
pa.fit(X\_train, y\_train)  
pa\_acc = accuracy\_score(y\_test, pa.predict(X\_test))  
print(f"Passive Aggressive Classifier Accuracy: {pa\_acc:.2%}")

Passive Aggressive Classifier Accuracy: 86.52%

#### 18. Bagging Classifier

from sklearn.ensemble import BaggingClassifier  
bag = BaggingClassifier(random\_state=42)  
bag.fit(X\_train, y\_train)  
bag\_acc = accuracy\_score(y\_test, bag.predict(X\_test))  
print(f"Bagging Classifier Accuracy: {bag\_acc:.2%}")

Bagging Classifier Accuracy: 85.82%

#### 19. Voting Classifier (Soft Voting)

from sklearn.ensemble import VotingClassifier  
voting = VotingClassifier(estimators=[  
 ('lr', lr), ('rf', rf), ('knn', knn)  
], voting='soft')  
voting.fit(X\_train, y\_train)  
voting\_acc = accuracy\_score(y\_test, voting.predict(X\_test))  
print(f"Voting Classifier Accuracy: {voting\_acc:.2%}")

Voting Classifier Accuracy: 87.94%

#### 20. CatBoost Classifier

from catboost import CatBoostClassifier  
cat = CatBoostClassifier(verbose=0, random\_state=42)  
cat.fit(X\_train, y\_train)  
cat\_acc = accuracy\_score(y\_test, cat.predict(X\_test))  
print(f"CatBoost Accuracy: {cat\_acc:.2%}")

CatBoost Accuracy: 88.65%

## Model Comparison

all\_scores = {  
 'KNN': knn\_acc,  
 'Random Forest': rf\_acc,  
 'Decision Tree': dt\_acc,  
 'Logistic Regression': lr\_acc,  
 'ANN (MLP)': ann\_acc,  
 'Naive Bayes': nb\_acc,  
 'Extra Trees': et\_acc,  
 'Gradient Boosting': gb\_acc,  
 'AdaBoost': ada\_acc,  
 'XGBoost': xgb\_acc,  
 'LightGBM': lgb\_acc,  
 'Ridge': ridge\_acc,  
 'Lasso': lasso\_acc,  
 'SGD': sgd\_acc,  
 'LDA': lda\_acc,  
 'QDA': qda\_acc,  
 'Passive Aggressive': pa\_acc,  
 'Bagging': bag\_acc,  
 'Voting Classifier': voting\_acc,  
 'CatBoost': cat\_acc  
}  
  
  
best\_model = max(all\_scores, key=all\_scores.get)  
print(f"Best Model: {best\_model} with Accuracy: {all\_scores[best\_model]:.2%}")

**Best Model**: Random Forest Classifer with Accuracy: ~.89.36%

## Optimization Techniques

* **Genetic Algorithm (GA)**: Use DEAP to optimize feature subsets
* **Particle Swarm Optimization (PSO)**: Use pyswarm to tune hyperparameters

## Future Improvements

* Ensemble learning (Voting Classifier, Stacking)
* Hyperparameter tuning (GridSearchCV, RandomizedSearchCV)
* Advanced feature selection (FCBF, recursive elimination)
* Expand dataset or combine with clinical data if needed

## Importance of Machine Learning Models

Machine learning models like KNN, Random Forest, Decision Trees, SVM, and others play a crucial role in **data-driven decision-making**. They help in tasks such as classification, prediction, and pattern recognition across various domains, from **healthcare diagnostics** to **financial fraud detection**. By leveraging these algorithms, we can automate processes, extract insights from complex datasets, and improve accuracy in real-world applications, making AI-driven solutions more effective and accessible.

**Data Source**: Breast cancer somatic mutation dataset filtered to 358 samples.

**Source (Portal)**: [*https://portal.gdc.cancer.gov/projects/tcga-brca*](https://portal.gdc.cancer.gov/projects/tcga-brca)

**Target Task**: Predict patient survival outcome using somatic mutation profiles.