Machine Learning-Based Classification of Ovarian Cancer Using Gene Expression Data

# Abstract

This project explores the use of supervised machine learning algorithms to classify ovarian cancer samples based on gene expression profiles. Using a realistic synthetic dataset inspired by public gene expression databases (e.g., GSE26712), models were trained to distinguish between cancerous and non-cancerous ovarian tissue. Three classifiers—Random Forest, Support Vector Machine (SVM), and Logistic Regression—were implemented and evaluated on the basis of accuracy, confusion matrix, and ROC-AUC. The project demonstrates the potential of machine learning in assisting early cancer detection.

# Methods

The dataset comprised 60 samples with 100 gene features and a binary label indicating cancer presence (1) or absence (0). Data was split into training and testing sets (80:20 ratio). Feature selection was inherent to the Random Forest model, while Logistic Regression and SVM provided baseline comparisons. The models were evaluated on classification accuracy, ROC curves, and feature importance analysis. The pipeline was developed and executed in Python using Google Colab and libraries such as Pandas, scikit-learn, and Seaborn.

# Results

The Random Forest model achieved an accuracy of 75%, identifying key genes contributing to classification, including GENE10, GENE25, and GENE40. Logistic Regression and SVM showed comparable performance, with ROC-AUC curves illustrating the trade-off between true positive and false positive rates. Confusion matrices provided insights into prediction reliability, while bar plots of feature importances highlighted gene contributions.

# Conclusion

This project demonstrates the feasibility of using machine learning for classifying ovarian cancer based on gene expression data. Although trained on simulated but biologically inspired data, the models show promise for future application on clinical datasets. This work represents a foundational step in integrating computational approaches into cancer diagnostics and lays the groundwork for further bioinformatics research.

# Appendix: Detailed Notebook Explanation

Detailed Explanation: Ovarian Cancer ML Notebook

Cell 1 (Markdown): # 🧬 Ovarian Cancer Classification - Upgraded ML Pipeline

This notebook uses real gene expression data to classify ovarian cancer using multiple ML models and evaluates them using accuracy, confusion matrix, and ROC-AUC.

Cell 2 (Code):  
# 📦 Import Libraries  
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  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.linear\_model import LogisticRegression  
from sklearn.svm import SVC  
from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, roc\_curve, auc

%matplotlib inline

Explanation:  
This cell imports essential Python libraries:   
- `pandas` and `numpy` for data manipulation  
- `matplotlib.pyplot` and `seaborn` for plotting  
- `sklearn` for machine learning models and evaluation metrics

Cell 3 (Code):  
# 📂 Upload Dataset  
from google.colab import files  
uploaded = files.upload()

import io  
df = pd.read\_csv(io.BytesIO(uploaded['realistic\_ovarian\_gene\_expression.csv']))  
df.head()

Explanation:  
This cell lets the user upload the dataset from their local system using Google Colab's file upload widget.

Cell 4 (Code):  
# 📊 Class Distribution  
sns.countplot(data=df, x='Cancer')  
plt.title('Class Distribution (0=Normal, 1=Cancer)')  
plt.show()

Explanation:  
This cell plots a bar chart showing how many samples belong to each class (cancer or normal). Useful to check for class imbalance.

Cell 5 (Code):  
# 🧪 Train/Test Split  
X = df.drop('Cancer', axis=1)  
y = df['Cancer']

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

Explanation:  
This cell splits the dataset into training and testing sets. 80% of data is used for training, 20% for testing.

Cell 6 (Code):  
# 🌲 Random Forest  
rf\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)  
rf\_model.fit(X\_train, y\_train)  
rf\_pred = rf\_model.predict(X\_test)  
rf\_proba = rf\_model.predict\_proba(X\_test)[:, 1]

print("Random Forest Accuracy:", accuracy\_score(y\_test, rf\_pred))  
print(classification\_report(y\_test, rf\_pred))

Explanation:  
Trains a Random Forest model on the training data. `predict()` gives the class prediction, while `predict\_proba()` is used for ROC curve.

Cell 7 (Code):  
# 🔁 Logistic Regression  
logreg\_model = LogisticRegression(max\_iter=1000)  
logreg\_model.fit(X\_train, y\_train)  
logreg\_pred = logreg\_model.predict(X\_test)  
logreg\_proba = logreg\_model.predict\_proba(X\_test)[:, 1]

print("Logistic Regression Accuracy:", accuracy\_score(y\_test, logreg\_pred))  
print(classification\_report(y\_test, logreg\_pred))

Explanation:  
Trains a Logistic Regression model on the same dataset and evaluates performance.

Cell 8 (Code):  
# 💠 Support Vector Machine  
svm\_model = SVC(kernel='linear', probability=True)  
svm\_model.fit(X\_train, y\_train)  
svm\_pred = svm\_model.predict(X\_test)  
svm\_proba = svm\_model.predict\_proba(X\_test)[:, 1]

print("SVM Accuracy:", accuracy\_score(y\_test, svm\_pred))  
print(classification\_report(y\_test, svm\_pred))

Explanation:  
Trains a Support Vector Machine (SVM) with a linear kernel and probability estimation enabled.

Cell 9 (Code):  
# 📉 Confusion Matrix for Random Forest  
cm = confusion\_matrix(y\_test, rf\_pred)  
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')  
plt.xlabel('Predicted')  
plt.ylabel('Actual')  
plt.title('Random Forest - Confusion Matrix')  
plt.show()

Explanation:  
Plots a heatmap showing the confusion matrix for the Random Forest predictions. Helps visualize true vs. predicted classes.

Cell 10 (Code):  
# 📈 ROC Curves  
fpr\_rf, tpr\_rf, \_ = roc\_curve(y\_test, rf\_proba)  
fpr\_lr, tpr\_lr, \_ = roc\_curve(y\_test, logreg\_proba)  
fpr\_svm, tpr\_svm, \_ = roc\_curve(y\_test, svm\_proba)

plt.figure(figsize=(8, 6))  
plt.plot(fpr\_rf, tpr\_rf, label='Random Forest (AUC = {:.2f})'.format(auc(fpr\_rf, tpr\_rf)))  
plt.plot(fpr\_lr, tpr\_lr, label='Logistic Regression (AUC = {:.2f})'.format(auc(fpr\_lr, tpr\_lr)))  
plt.plot(fpr\_svm, tpr\_svm, label='SVM (AUC = {:.2f})'.format(auc(fpr\_svm, tpr\_svm)))  
plt.plot([0, 1], [0, 1], 'k--')  
plt.xlabel('False Positive Rate')  
plt.ylabel('True Positive Rate')  
plt.title('ROC Curves')  
plt.legend()  
plt.show()

Explanation:  
Generates and plots the ROC curves for all three models (RF, SVM, LogReg), showing how well each model separates the classes.

Cell 11 (Code):  
# 🌟 Feature Importance (Random Forest)  
importances = rf\_model.feature\_importances\_  
features = X.columns

# Sort top 10 genes  
indices = np.argsort(importances)[-10:]  
plt.figure(figsize=(8, 6))  
sns.barplot(x=importances[indices], y=features[indices])  
plt.title('Top 10 Important Genes (Random Forest)')  
plt.xlabel('Importance')  
plt.ylabel('Gene')  
plt.show()

Explanation:  
Plots the top 10 most important genes based on Random Forest feature importance scores. Indicates which genes influenced the model most.