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, tes

# 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_sta
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, randc
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', ra
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=
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 QuadraticDiscriminantAnalysi
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
```

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