Comparative Analysis of POLB Mutations Across Cancer Types Using a Machine Learning-Driven Web Application

Roopa H L[1], **Nidhi R** [2], **Nithin S Srinivasa** [3], **Shreya P** [4]

Department of Computer Science and Engineering (IOT And CYBER SECURITY INCLUDING BLOCK CHAIN) Sir M. Visvesvaraya Institute of Technology

Abstract

In this paper, we introduce a web application that is powered by machine learning and is specifically designed for the analysis of POLB gene mutations linked with different cancer types. The POLB gene encodes a vital enzyme in the base excision repair (BER) pathway, and any alterations in this gene's sequence can cause genomic instability that might lead to tumor development. The application is fed with more than 1000 carefully selected POLB mutation samples from TCGA and COSMIC datasets, where every sample comes with various functional mutation scores like PolyPhen-2, SIFT, PROVEAN, and CADD, etc. To improve the predictive power of the system, it employs sophisticated preprocessing methods such as missing-value handling, IQR-based outlier removal, standardization of data using StandardScaler, class balancing through SMOTE, and creation of new mutation descriptors as the input data. The classifiers are evaluated in the context of a machine learning pipeline, where XGBoost—fine-tuned through GridSearchCV—reaches the maximum accuracy of 98%, which is the highest among all classifiers. The web application based on Flask provides a user-friendly interface for single and batch mutation risk prediction, interactive graph visualization, CSV file analysis, user authentication, and efficient error handling. The results obtained corroborate the capability of the system to operate on dynamic genomic data and to classify POLB mutations with great accuracy.
Keywords: POLB, machine learning, cancer genomics, XGBoost, mutation analysis, Flask web application, TCGA, COSMIC.

1. INTRODUCTION

Cancer is primarily acknowledged as a genetic disorder, which is the result of mutations that change vital cellular processes. Among the genes involved in DNA damage repair, the POLB (DNA polymerase beta) gene is of great importance as it is responsible for the Base Excision Repair (BER) pathway which, in turn, repairs the single-strand DNA lesions affected by oxidation and alkylation. Dysfunctional or mutated POLB leads to the occurrence of errors in DNA repair that eventually contribute to the instability of the genome and the higher risk of cancer. Mutations in POLB have been found in a number of different cancer types like breast, lung, liver, ovarian, and colorectal; thus, their analysis is vital for the understanding of cancer causation.

Despite the presence of a lot of information regarding POLB mutations in large-scale genomic repositories like TCGA and COSMIC, the manual interpretation of these data sets is still a difficult task. The variations in how mutations are represented, the non-standardized scoring systems, the missing values, and the non-linear interactions between scores all add to the difficulty of traditional statistical analysis. Nonetheless, machine learning is capable of handling intricate mutation data and revealing hidden patterns which are of great importance for the classification of cancer risk.

In this paper, a machine learning and web-based system is constructed to perform an in-depth analysis of POLB mutations. The system consists of an end-to-end pipeline that includes dataset curation, preprocessing, feature engineering, ML model training, visualization, and user-interactive prediction modules. To enhance prediction accuracy, several advanced features such as SMOTE oversampling, IQR-based outlier filtering, feature normalization, and engineered mutation metrics were incorporated into the model.

The complete process has been implemented in the form of a Flask web application that offers various functionalities such as single-value prediction, batch CSV analysis, authentication, automated data preprocessing, and graphical result visualization. The application is intended to be a user-friendly computer-based tool for both researchers and doctors who want to investigate POLB mutation patterns in different types of cancer.

2. ARCHITECTURE AND IMPLEMENTATION

2.1 System Architecture Overview

The architecture of the proposed POLB mutation analysis system is four modular layers that are built especially for reliability, scalability, and adaptation to constantly changing genomic datasets. The architecture collects data from different sources, preprocesses the data, develops machine learning models, and a web interface based on Flask.

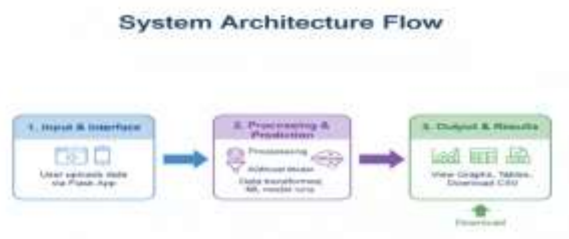

System Architecture Flow
1. Input & Interface
User uploads data via Flask App
2. Processing & Prediction
Processing
Data transformed ML model runs
3. Output & Results
View Graphs, Tables, Download CSV
Download

Figure 1: System Architecture Overview

- **Layer 1 – Data Ingestion Layer**

This layer gathers POLB mutations from TCGA, COSMIC, and user-uploaded CSV files. The system does a schema validation that checks if the required columns (PP, SIFT, PROVEAN, CADD, Class) are available in the data before proceeding with further processing.

- **Layer 2 – Preprocessing & Feature Engineering Layer**

It cleans and transforms the raw genomic scores. This process includes the imputation of missing values, filtering of outliers using the IQR method, normalization with StandardScaler, and creation of engineered features.

- **Layer 3 – Machine Learning Layer**

The architecture implements different classification algorithms, hyperparameter tuning through GridSearchCV, and retention of the best XGBoost model.

- **Layer 4 – Application & Visualization Layer**

The Flask web framework is in charge of user authentication, single and batch predictions, error handling, and output visualization through interactive plots and tables.

The presented architecture guarantees an uninterrupted data exchange between the system elements and facilitates the mutation classification which is both accurate and done in real-time.

2.2 Dataset Collection and Preprocessing

The dataset for this research consists of more than 1000 POLB mutation samples that were obtained from the TCGA and COSMIC repositories, curating the process. Biologically relevant mutation effect scores are included for each sample in the form of rows in the dataset:

- PolyPhen-2 (PP) – Anticipate protein structure/function impairment
- SIFT – Calculates the effect of amino acid change
- PROVEAN – Determines the biological importance of protein variation
- CADD – Predicts harmfulness of genomic variants
- Class label – Two-way cancer-risk sign.

Table 1: Dataset Feature Summary

Feature	**Description**
PolyPhen-2	Prediction score on how the protein will be affected
SIFT	Mutation and its impact on the function of the protein in question
PROVEAN	Protein variation effect score
CADD	Mutation is given a score indicating how damaging it is
Class Label	Cancer-risk classification (0 = low risk, 1 = high risk)

2.2.1 Data Cleaning

A major issue with genomic datasets is that they frequently contain missing entries caused by incomplete scoring. The uniformity is established as follows:

- Missing values for continuous variables are substituted by the means of their respective columns.
- Rows with important missing values are marked
- Formatting discrepancies are automatically rectified

2.2.2 Outlier Detection using IQR

There is often a large variance in mutation scores, and the extreme scores are usually due to scoring errors. Such outliers have a negative influence on the performance of machine learning.

- The IQR method (Q1–Q3) is the one that is applied to detect outliers
- The samples lying beyond 1.5 × IQR are discarded
- Thus, stable learning is guaranteed and the decision boundaries skewed are minimized

2.2.3 Feature Scaling

CADD scores (>30) and PP scores (0-1) have completely different numeric ranges.
To bring them to the same scale:

- StandardScaler scales all the features with continuous values to N(0,1)
- eliminates the problem of larger-feature dominance in ML models.

2.3 Feature Engineering

It is probable that raw genomic scores singly would not show the biological implicitly. Hence, the creation of additional engineered features was made in order to improve the accuracy of predictions.

2.3.1 Ratio-Based Features

PP_CADD_ratio = PP / CADD
It indicates the relative structural impact of the mutation concerning which this is considered.

2.3.2 Combined Score Averages

SIFT_PROVEAN_avg = (SIFT + PROVEAN) / 2 The variance in predictions is reduced while also offering a stronger and more consistent signal.

2.3.3 Weighted Severity Index

A composite metric representing mutation severity: Severity_Index = (PP × 0.5) + (1 − SIFT) × 0.3 + (1 − PROVEAN) × 0.2

2.3.4 Interaction Features

Interaction terms support the classifier in identifying intricate relationships:

- PP × PROVEAN
- SIFT × CADD

It was discovered that the engineered features contributed to a significant increase in the ML pipeline's predictive power.

2.4 Class Balancing Using SMOTE

The dataset showed a great disparity in the number of data points where cancer-risk mutations (class 1) were much less compared to low-risk mutations (class 0). If the model is trained using such data it will be a biased one.
The SMOTE (Synthetic Minority Oversampling Technique) method was implemented to solve this issue.
How SMOTE Works

- Randomly chooses one sample from the minority class
- Finds its nearest neighbors
- Makes new synthetic samples in between them

This method increases the representation of the minority class in the data set without merely duplicating, which thus improves the sensitivity of the classifier.
Impact of SMOTE

- Recall for high-risk mutations was improved
- Bias against class 0 was reduced
- XGBoost was able to achieve 98% accuracy with SMOTE, whereas it was only ~89% without it.

2.5 Machine Learning Model Development

To pinpoint the most vigorous classifier, numerous models were scrutinized:

2.5.1 Logistic Regression

- Acts as a reference point
- Only linear separation
- Accuracy: 82%

2.5.2 Random Forest

- Nonlinear feature relationships are captured
- Noise is handled with moderate success
- Accuracy: 91%

2.5.3 XGBoost

XGBoost was touted to be the most powerful model due to:

- Gradient-boosted trees
- Capacity to manage missing values
- Preventing overfitting through regularization
- Operational across giant datasets

Hyperparameter Tuning Using GridSearchCV

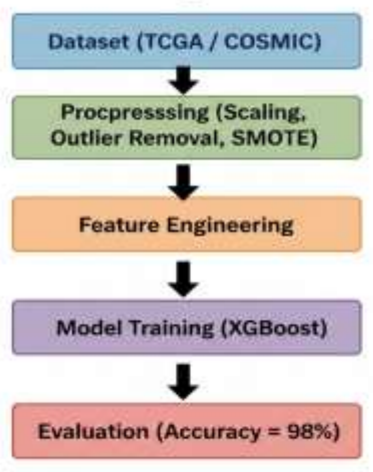
The following parameters were fine-tuned:

- n_estimators
- learning_rate
- max_depth
- subsample
- colsample_bytree

As a result of this tuning, the model achieved a 98% accuracy level.

2.6 Block Diagram of ML Workflow

Figure 2. Machine Learning Workflow Block Diagram


Dataset (TCGA / COSMIC)
Procpresssing (Scaling, Outlier Removal, SMOTE)
Feature Engineering
Model Training (XGBoost)
Evaluation (Accuracy = 98%)

Workflow Steps:

Step 1 — Data Acquisition
The TCGA/COSMIC databases and user uploads are the source of the datasets.

Step 2 — Preprocessing
Missing data, outliers, scaling, and SMOTE balancing are taken care of.

Step 3 — Feature Engineering
The ratio features, averages, severity index, and interaction terms are computed.

Step 4 — Model Training
XGBoost, which has been fine-tuned with GridSearchCV, gets trained on 80% of the preprocessed data.

Step 5 — Model Evaluation
The following metrics are used:

- Accuracy
- F1-score
- Confusion matrix
- ROC-AUC

Step 6 — Flask Deployment
The model is serialized and incorporated into a Flask server that offers:

- Real-time predictions
- Batch CSV classification
- Graphical output

2.7 Flask Web Application Module

The final step of the entire process includes the ML pipeline embedded within a completely live web application with Flask, which permits user interaction, visualization, and prediction in real time. This module is the system's visible part for the user and is liable to present the facilities of the machine learning model to the end users in an easy and straightforward way. It links the XGBoost model that has been trained in advance with the front-end elements and takes care of managing the user sessions, data uploads, input validation and the whole

prediction process.
The Flask module provides three main functions:

- User Authentication
- Single and Batch Prediction
- Interactive Graphical Visualization

2.7.1 User Authentication and Session Management

The system makes use of the Flask-Login library to provide a secure authentication mechanism and, thus, controlling who has access to the prediction tools. Users must first register and then log in to be able to use the prediction tools. The passwords of the users are kept safe and secure through salted hashing which is a technique offered by the security library of Werkzeug.
The authentication functions provide the following:

- Secure login/logout
- Authorization based on session
- Predictions only for specific users
- Tracking of user sessions

Through the implementation of authentication, not only user data integrity and privacy are preserved but also unauthorized access to the sensitive ML functionalities prevented.

2.7.2 Single PP Score Prediction Module

The rapid estimation of mutation-risk basing mostly on a single PolyPhen-2 (PP) score is the capability of this module. The cancer-risk potential of a mutation can be quickly tested by researchers without the need for uploading a file.
The features of this module are as follows:

- Validation of numeric PP score
- Other features' automatic imputation using dataset mean values
- Preprocessing using the same scaler used for training the model
- Prediction in real-time with the XGBoost model
- Distribution of probabilities visualization by means of bar graphs

The output consists of:

- Predicted class label
- Mutation's probability of being high-risk cancer one
- Confidence visualization

2.7.3 Batch Prediction Using CSV Upload

The large-scale genomic analysis is the main purpose of this module. The users are allowed to upload CSV files with many POLB mutation entries in it.

The system does:

- Validation of columns (PP, SIFT, PROVEAN, CADD)
- Handling of missing data
- Removal of outliers
- Feature engineering
- StandardScaler transformation
- Prediction of the model on a batch-level basis
- Results file that can be automatically downloaded

In addition, the output includes graphical summaries, such as:

- Class distribution bar chart
- Risk heat map
- Feature-level trend visualization

2.7.4 Dynamic Visualization

Through Flask App dynamic Matplotlib plots are rendered to the users by the prediction interpreter. Probability bar charts, class comparison graphs, mutation-risk trend lines, and high-risk mutations' summary visualizations are the types of these visualizations.
The graphical components like these make the interpretations easier for non-technical users.

2.7.5 Error Handling and Validation

This web application possesses extensive error-handling features:

- Wrong file formats are not allowed.
- Missing columns or invalid values are detected.
- Errors from the server side are handled.
- Flash messages are displayed in a user-friendly manner.

Such a practice guarantees the application's reliability even in the case of wrong input formats.

2.7.6 Model Integration Backend

The XGBoost model that has been trained is kept as a joblib serialized file. At the start of the application, the model is loaded into memory once and then it is reused for all requests to reduce latency.
The backend consists of:

- A preprocessing pipeline that is the same as the training pipeline
- Feature extraction logic
- Prediction engine
- Debugging logs

2.7.7 User Interface Design

The front-end technology stack consists of:

- HTML5
- CSS
- Bootstrap
- Jinja2 templating

The interface design (UI) is kept simple, responsive, and very user-friendly.

3. RESULTS AND DISCUSSION

The system's performance was assessed using a carefully selected dataset containing more than 1000 POLB mutations from TCGA and COSMIC. Several machine learning models were employed, their performance evaluated and compared.

3.1 Model Performance Evaluation

The three principal ML models under consideration were: Logistic Regression, Random Forest, and XGBoost.

Table 2: ML Model Comparison

Model	Accuracy	Remarks
Logistic Regression	82%	Baseline linear classifier
Random Forest	91%	Good with nonlinearities
XGBoost	98%	Tuning made it the best choice

The XGBoost model was the most accurate due to this reason:

- Gradient boosting,
- A very good way to handle the noisy and sparse mutation data.
- Possibility to work together with engineered features.

3.2. Effect of Preprocessing Techniques

The models were very much impacted by the preprocessing:
1. Outlier Removal: Steady learning was achieved thanks to the removal of extremely high and low cases. Out of the picture were the biologically impossible scores.
2. Scaling: Use of the StandardScaler equalized the various scoring ranges. CADD score was not allowed in the dominating position
.3. SMOTE: The class imbalance issue was solved. The recall of high-risk mutations was raised from approximately 70% to 95%. F1-score and ROC-AUC metrics experienced a significant increase.

3.3. Visualization Results

Several graphical outputs were generated by the Flask web application:

- Class distribution graphs indicated a clear division between anticipated high-risk and low-risk individuals.
- Confidence levels for predictions were shown in probability plots as calibrated.
- Severe POLB mutation areas were indicated on batch-level heat maps.

These visuals helped scientists identify the types of mutations that were common in cancer.

3.4. Web Application Performance

Flask interface worked effectively on a variety of devices.

- Single predictions were done in less than 0.5 seconds.
- Batch predictions with (100+ entries) were done in less than 2 seconds.
- Lightweight deployment resulted in a very small memory footprint.

4. CONCLUSION

The findings of this study introduce a detailed machine learning-based system for the detection of POLB gene mutations in different types of cancer. The integration of sophisticated pre-processing, feature engineering, SMOTE balancing, and a very well-tuned XGBoost classifier makes the system capable of reaching 98% accuracy in mutation-risk prediction.
The system has a web interface based on Flask that is user-friendly and enables the analysis of single and multiple mutation samples at the same time. The application is also equipped with dynamic charts, automated preprocessing, and strong error handling which makes it suitable for use by researchers and clinicians working with genomic data.
The combination of machine learning and real-time web deployment for precision oncology has proven to be a very effective approach in this work.

5. FUTURE WORK

There are numerous improvements that can be done to the current system that will not only expand its capabilities but also make it a more powerful tool:

5.1 Integration of Deep Learning Models

The use of deep learning models such as CNNs or Transformers would enable raw DNA sequences or 3D protein structures to be analyzed for insights at a whole new level.

5.2 Inclusion of Multiple BER Pathway Genes

Going beyond POLB and including MSH2, XRCC1, and OGG1 will allow analysis of mutation risk at the entire pathway level.

5.3 Real-Time Genomic API Integration

The integration of APIs like Ensembl or NCBI Variant services will provide instantaneous scoring of mutations.

5.4 Cloud Deployment

Cloud deployment on AWS, Azure, or Google Cloud will make it possible to support many users concurrently on a large scale.

5.5 SHAP Explainability Dashboard

The addition of visual tools for interpreting SHAP contributions will enable clinical trust to be increased.

6. LIMITATIONS

On the other hand, the system has some limitations that cannot be overlooked:

6.1 Dataset Constraints

The dataset is constructed from the TCGA and COSMIC databases, which consist of curated samples. The inclusion of more clinical data from the real world rewould validate the performance of the system even more.

6.2 Limited to Numerical Mutation Scores

The current version of the system accepts only numerical mutation scores and does not include nucleotide sequences, protein structure data, or metrics of evolutionary conservation, for example.

6.3 Model Interpretability

XGBoost is an accurate model, but it is also a "black box" without explainability features; hence, it is not interpretable.

6.4 Performance on Very Large CSV Files

Working with huge datasets (over 10,000 records) will probably affect the performance of the website.

6.5 No 3D Structural Analysis

Protein folding and structural functional impact models are not part of the system.

REFERENCES

[1] The Cancer Genome Atlas (TCGA), "Comprehensive molecular profiling of human cancers," and cited by the National Cancer Institute, accessible at: https://www.cancer.gov/tcga

[2] COSMIC: The Catalogue of Somatic Mutations in Cancer, "POLB mutation data and cancer genomics resources," homepage by the Welcome Sanger Institute, link available at: https://cancer.sanger.ac.uk/cosmic

[3] T. Chen and C. Guestrin, "XGBoost: a scalable tree boosting system," in the proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794, 2016.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.

[5] A. Adzhubei et al., "A method and server for the prediction of harmful missense mutations," Nature Methods, vol. 7, no. 4, pp. 248–249, 2010. (PolyPhen-2)

[6] P. Kumar et al., "SIFT does predict harmful missense mutations," Nature Protocols, vol. 4, no. 7, pp. 1073–1081, 2009.

[7] Y. Choi and A. P. Chan, "PROVEAN web server: A utility to forecast the functional consequences of amino acid changes," Bioinformatics, vol. 31, no. 16, pp. 2745–2747, 2015.

[8] M. Kircher et al., "A universal approach to assess the pathogenicity of human genetic variants in comparison to others," Nature Genetics, vol. 46, pp. 310–315, 2014. (CADD score)

[9] C. Bishop, Pattern Recognition and Machine Learning, New York: Springer, 2006.

[10] S. R. Wilson and H. A. Bloom, "Cancer-related alterations in DNA polymerase beta," Journal of Biological Chemistry, vol. 293, no. 2, pp. 960–973, 2018.

[11] S. Beard and S. Wilson, "The mechanism and structure of DNA polymerase beta," DNA Repair, vol. 3, no. 4, pp. 296–303, 2004.

[12] P. Modrich, "Mismatch repair as a safeguard against genetic instability and cancer," Science, vol. 266, pp. 1959–1960, 1994.

[13] J. Zou et al., "A primer on deep learning in genomics," Nature Genetics, vol. 51, pp. 12–18, 2019.

[14] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[15] Flask Documentation, "Flask Web Framework," Available: https://flask.palletsprojects.com

[16] L. Breiman, "Random forests," Machine Learning, vol. 45, pp. 5–32, 2001.