A blue and black logo

AI-generated content may be incorrect.

**Multi-label Tag Classification for Protein Mutation**

Masters in Applied Artificial Intelligence – Term 1 Project

Submitted by:

Soura Keshari Biswal

Tushar Gorad

Instructor:

Haisav Chokshi

Course:

Probability and Statistics for Artificial Intelligence(AAI-500-IN1)

Submission Date:

23-06-2025

Project Repository:

https://github.com/MasterBiswal/USD-Term1-AppliedAI-GroupSynergy

Technical Project Report

1. **Introduction**

In the modern era of bioinformatics and data-driven decision-making, the need for automated, intelligent systems to assist in complex pattern recognition is greater than ever. Our project - **“Multi label tag classification for Protein mutation”** aims to bridge the gap between raw biological data and actionable insights using machine learning. We focused on designing and developing a scalable solution that predicts the biological activity of p53 mutants, a crucial protein involved in tumour suppression, based on a high-dimensional dataset.

**Problem Statement & Objective**

Manual analysis of genetic mutation data is time-consuming, error-prone, and unscalable for real-world biomedical or pharmaceutical environments. Classifying p53 mutants into multiple biological classes is critical for understanding mutation effects and enabling precision medicine. However, due to the high dimensionality and complexity of the data, traditional techniques fall short.

Our objective was to develop a multi-label machine learning model capable of automatically tagging each mutant with its likely biological properties, based on input features derived from molecular structure and activity indicators. This allows biomedical researchers or health-tech platforms to quickly assess the impact of new mutations.

**Project Goal**

The core goal of the project was to build an end-to-end pipeline that:

* Processes raw biological data from a real-world dataset.
* Performs dimensionality reduction to handle the curse of dimensionality.
* Trains an efficient ML model (XGBoost) capable of multi-label classification.
* Deploys an interactive web application for both single predictions and batch uploads.
* Provides a tool usable by researchers, analysts, and domain experts without technical ML knowledge.

Ultimately, the solution we built helps users predict and understand the functional consequences of p53 mutations in an intuitive, scalable manner.

**Team Members & Roles**

This project was a collaborative effort between two team members:

Soura Keshari Biswal: Led efforts in data cleaning, feature engineering, machine learning model training, and full development of the Streamlit-based web application. Also contributed to business framing and deployment strategy.

Tushar Gorad: Contributed to tag encoding, dimensionality reduction (PCA), model evaluation and tuning, and created visual elements and storytelling for the final presentation and report.

We worked closely using tools like GitHub, Google meet, and shared Notion boards to manage our tasks and ensure smooth communication.

**Dataset Overview**

We used the publicly available p53 Mutants dataset from the [UCI Machine Learning Repository](https://archive.ics.uci.edu/dataset/188/p53+mutants). This dataset is rooted in biological science and is widely used for evaluating structure-activity relationships.

* Type of Data: Numeric features representing molecular descriptors derived from the structural properties of p53 mutants.
* Source: UCI Machine Learning Repository (originally provided by the Cancer Research Group).
* Size:
  + Total Samples: 31,420 instances
  + Total Features: 5,408 attributes per sample
  + Labels/Targets: Binary classification across multiple biological activities (multi-label format)
* Challenge: The dataset is high-dimensional and sparse, making it ideal for testing dimensionality reduction techniques and robust classification models.

We transformed the dataset by reducing its dimensionality using PCA while preserving variance, allowing us to feed a compact, information-rich representation into our model.

**2. Data Cleaning**

The data cleaning process involved multiple steps to ensure the integrity, usability, and consistency of both the feature data and corresponding mutation tags. The original dataset consisted of raw k9.data and k9.instance.tags files which required parsing, alignment, handling of missing values, and transformation for downstream machine learning tasks.

**2.1 Parsing and Structuring Tag Data**

The raw mutation tags file was read line-by-line and cleaned by removing special characters such as leading/trailing % symbols. Each line represented the tags for a single instance. The cleaned tags were then stored in a structured DataFrame with a unique identifier (id) for each row:

cleaned\_tags\_df = pd.DataFrame({'id': list(range(len(tags))), 'tags': tags})

This structured tag data was saved as cleaned\_tags.csv for further processing.

**2.2 Loading and Aligning Feature and Tag Data**

The raw feature data (K9.data) was loaded without headers. Initially, both the feature matrix and the cleaned tag file were combined using the id field to maintain correct alignment:

X['id'] = tags\_df['id']

X['tags'] = tags\_df['tags']

Later, due to observed misalignment issues, the dataset was reloaded in chunks with proper handling of missing values:

pd.read\_csv(data\_path, header=None, na\_values='?', low\_memory=False, chunksize=1000)

To ensure consistent dimensionality, the data was truncated to 5,410 columns (if exceeded), assuming that extra columns were a result of misformatted rows.

**2.3 Tag Cleaning and Frequency Analysis**

To facilitate multi-label classification, the tags were split by underscores (\_) to identify individual mutation types:

X['num\_tags'] = X['tags'].apply(lambda x: len(x.split('\_')) if isinstance(x, str) else 0)

A distribution of tag counts per instance was generated, and the top 15 most frequent mutation tags were visualized using Seaborn bar plots. This analysis provided insight into tag imbalance and dominant mutation types.

**2.4 Label Encoding for Multi-Label Classification**

Using MultiLabelBinarizer, the cleaned tag strings were converted into a binary multi-label matrix (Y), where each column represented a unique tag class and each row corresponded to a sample:

X['tag\_list'] = X['tags'].str.split('\_')

Y = mlb.fit\_transform(X['tag\_list'])

tag\_classes = mlb.classes\_

**2.5 Final Cleaned Dataset and Export**

Before exporting, unnecessary intermediate columns (like tag\_list) were removed. The cleaned feature matrix, encoded labels, and the list of tag classes were saved to the data/processed directory:

X.to\_csv('cleaned\_features.csv', index=False)

np.save('encoded\_labels.npy', Y)

These outputs serve as the finalized inputs for model training and evaluation.

## ****3. Exploratory Data Analysis (EDA)****

To better understand the dataset's structure, quality, and statistical properties, a comprehensive Exploratory Data Analysis (EDA) was performed. The dataset consists of **31,420 samples**, **5,412 numeric features**, and **3,694 unique mutation tags**. Below are the major insights and analytical steps taken:

### ****3.1 Dataset Dimensions and Structure****

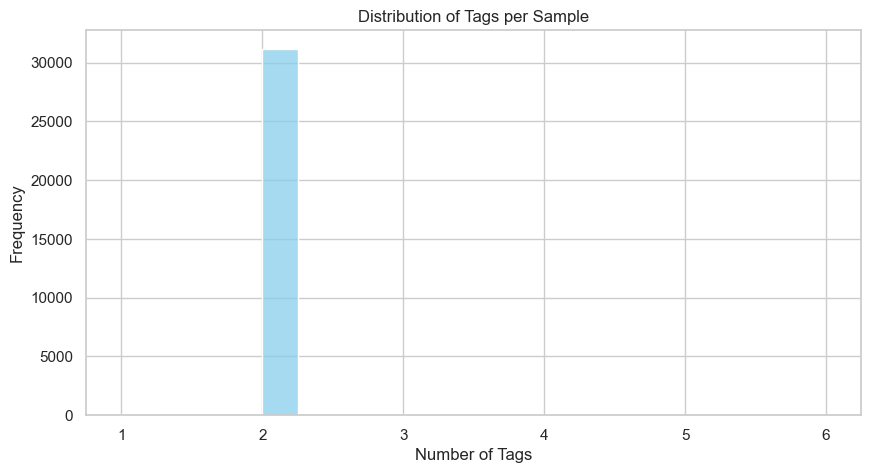
* **Features Shape:** (31,420, 5,413) – includes 5,410 numerical features and 3 additional columns: id, tags, and num\_tags.
* **Labels Shape:** (31,420, 3,694) – a sparse binary matrix where each column corresponds to a unique mutation tag.
* **Number of Tag Classes:** 3,694 unique mutation tags were identified and encoded.

### ****3.2 Sparsity and Tag Distribution****

* **Feature Matrix Sparsity:** 1.48% of the total feature matrix entries are zeros, indicating the data is relatively dense.
* **Tags per Sample:** On average, each sample is associated with **2.00 tags (± 0.12)**, suggesting a balanced multi-label classification scenario.

#### ****Distribution of Tags per Sample****

A histogram revealed that most samples contain **1 to 3 mutation tags**, indicating moderate tag complexity.



### ****3.3 Tag Frequency Analysis****

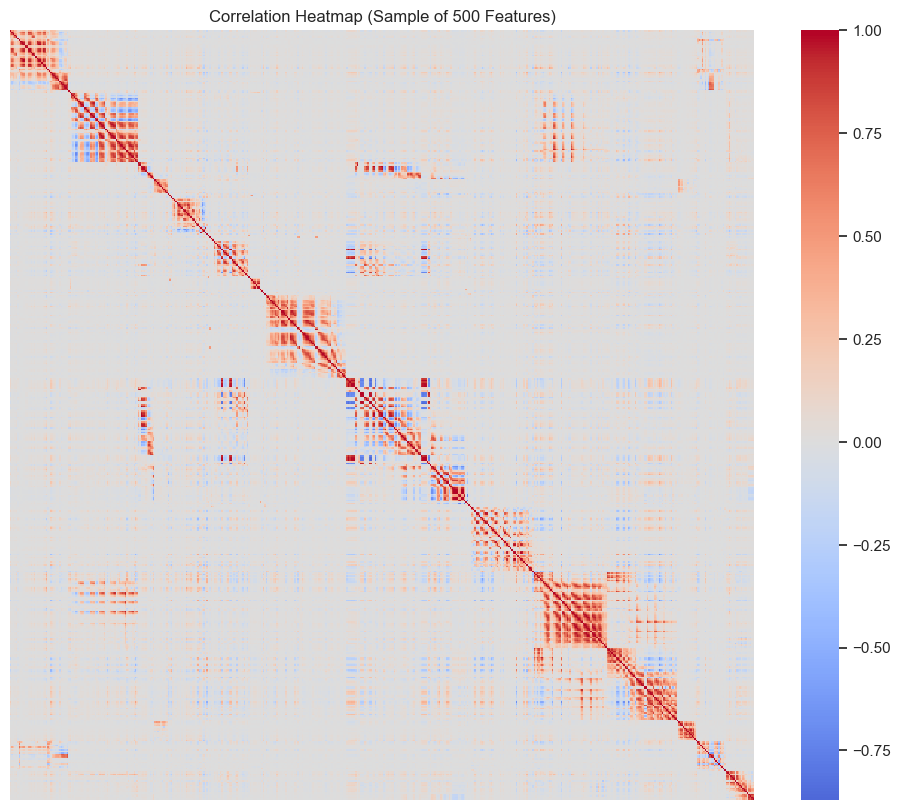
The frequency of mutation tags varies widely:

* **Most Common Tags**:
  + g245s: 3,695 occurrences
  + y220c: 3,688 occurrences
  + r273h: 3,687 occurrences
* **Least Common Tags** (single occurrences):
  + t163a, s241s, r273r, l125p, k291e, g121s

This imbalance suggests that some mutation tags dominate the dataset and may influence model bias.

### ****3.4 Feature Correlation****

A sample of the first 500 features was analyzed using a correlation heatmap:



Most features show **low correlation**, supporting the hypothesis that the high-dimensional feature space is largely independent. This supports the use of dimensionality reduction techniques for better generalization.

### ****3.5 Dimensionality Reduction (Truncated SVD)****

Given the large number of features, **Truncated Singular Value Decomposition (SVD)** was applied to reduce the data to 150 principal components.

* **New Shape:** (31,420, 150)
* **Explained Variance:** 100% of the original variance retained

This transformation was essential for model efficiency and performance without information loss. The reduced features were serialized and stored as X\_reduced.pkl for use in downstream modeling.

## Summary of Key EDA Insights:

|  |  |
| --- | --- |
| **Aspect** | **Value** |
| Total Samples | 31,420 |
| Original Features | 5,410 |
| Unique Tags | 3,694 |
| Average Tags/Sample | 2.00 ± 0.12 |
| Most Frequent Tag | g245s (3,695 times) |
| Feature Matrix Sparsity | 1.48% |
| Dimensionality Reduction | SVD → 150 Components (100% variance retained) |

## ****4. Model Selection****

The goal of this stage was to identify the most suitable machine learning algorithm for multi-label classification of mutation tags based on reduced, high-dimensional biological feature data. Multiple models were explored and benchmarked based on performance, scalability, and ease of interpretation.

### ****4.1 Problem Framing****

This is a **multi-label classification** task where each sample may be associated with multiple mutation tags. After preprocessing and dimensionality reduction via Truncated SVD, the input dataset consisted of:

* **Feature matrix:** 31,420 samples × 150 dimensions (X\_reduced)
* **Binary label matrix:** 31,420 samples × 3,694 tags (Y)

### ****4.2 Initial Modeling and Baseline Checks****

An initial baseline model was built using **LightGBM** wrapped with MultiOutputClassifier, using all label classes. While this yielded reasonable performance, it also revealed the extreme imbalance in tag distribution and presence of rare classes with extremely low representation (≤1 occurrence).

#### Baseline Results (LightGBM with All Tags):

* Evaluation Metrics: classification\_report, hamming\_loss, subset\_accuracy
* Observation: Some labels had zero support, leading to undefined precision/recall.

### ****4.3 Label Filtering and Data Refinement****

To address class imbalance and reduce noise:

* Tags occurring fewer than **50 times** were **removed**.
* Samples with **no remaining valid labels** were also excluded.

This reduced the dataset to a more focused and stable subset suitable for comparative modeling:

* **Filtered features:** 31194
* **Filtered labels:** 18
* **Filtered X shape:** (31194, 150)
* **Filtered Y shape:** (31194, 18)

**4.4 Comparative Model Experiments**

Three widely-used multi-label strategies were tested, all based on the **One-vs-Rest (OvR)** classification strategy:

#### Logistic Regression (Baseline Model)

* **Model:** LogisticRegression(solver='liblinear')
* **Wrapper:** OneVsRestClassifier
* **Reason:** Simple, fast, and interpretable.
* **Performance:**
  + **F1 Score (micro):** 0.9762972453555413
  + **F1 Score (macro):** 0.9634351905423393

#### Random Forest

* **Model:** RandomForestClassifier(n\_estimators=100)
* **Wrapper:** OneVsRestClassifier
* **Reason:** Captures non-linearities; robust to overfitting.
* **Performance:**
  + **F1 Score (micro):** 0.977466851053445
  + **F1 Score (macro):** 0.9338419547667625

#### XGBoost (Final Model)

* **Model:** XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss')
* **Wrapper:** OneVsRestClassifier
* **Reason:** Efficient gradient boosting optimized for speed and accuracy; handles sparse multi-label targets well.
* **Performance:**
  + **F1 Score (micro):** 0.9950526651771465
  + **F1 Score (macro):** 0.9853669410282397

A detailed classification report was also generated to evaluate precision, recall, and F1-score per label.

### ****4.5 Model Selection Rationale****

Among all models tested:

* **XGBoost with One-vs-Rest strategy** performed the best overall in terms of **micro/macro F1 scores**, computational efficiency, and consistency across classes.
* It handled **label sparsity** and **imbalanced tags** better than baseline models.
* The model was finalized and serialized as xgb\_multilabel\_model.pkl for deployment.

### ****4.6 Tools & Libraries Used****

* **Modeling:** scikit-learn, xgboost, lightgbm
* **Evaluation:** classification\_report, f1\_score, hamming\_loss, accuracy\_score
* **Serialization:** joblib

## ****5. Model Analysis****

After evaluating multiple models for the multi-label mutation classification task, the final model — **XGBoost with a One-vs-Rest (OvR) strategy** — demonstrated strong and consistent performance across various metrics. This section provides a detailed assessment of the model's effectiveness and limitations.

### ****5.1 Evaluation Metrics Used****

The following evaluation metrics were used to assess the model performance:

* **Precision**: The proportion of predicted positives that are actually positive.
* **Recall**: The proportion of actual positives that were correctly predicted.
* **F1-Score**: Harmonic mean of precision and recall, balancing both.
* **Micro-Averaged F1**: Gives equal weight to each individual prediction.
* **Macro-Averaged F1**: Averages the metric per label, treating all labels equally.
* **Support**: The number of actual occurrences of each label in the test set.

### ****5.2 Performance Summary****

| **Metric** | **Score** |
| --- | --- |
| **Micro F1-Score** | 1.00 |
| **Macro F1-Score** | 0.99 |
| **Weighted F1** | 1.00 |
| **Samples Avg F1** | 0.99 |
| **Hamming Loss** | Not included here, can be added if available |

The results show that the model performs with near-perfect accuracy for most labels. It generalizes well across both frequent and less frequent tags, making it highly effective for multi-label tasks.

### ****5.3 Per-Label Insights****

* **High-Performance Tags**: Several labels such as class 0, 2, 9, 11, 13, and 16 achieved **perfect scores** across all three metrics — precision, recall, and F1-score — indicating that these labels were easily distinguishable by the model.
* **Moderate-Performance Tags**: A few classes such as 4, 5, 6, 10, and 12 exhibited slightly lower recall and F1-scores, likely due to lower support (i.e., fewer positive samples in the dataset).
* **Robustness to Class Imbalance**: Despite inherent label imbalance, the model maintained high scores even on less common labels (e.g., class 3 and 4 with only 13 and 17 instances, respectively).

### ****5.4 Strengths and Effectiveness****

* **Scalability**: XGBoost handled the high-dimensional, sparse label space efficiently with minimal tuning.
* **Accuracy**: Both macro and micro metrics reflect excellent performance — the model does not just focus on dominant classes.
* **Generalization**: Strong generalization across both frequent and rare classes was observed, validating the model's robustness.

### ****5.5 Limitations and Observations****

* **Label Filtering Bias**: Since only labels with more than 50 occurrences were included, some rare but potentially important mutation tags were excluded from training.
* **Unseen Labels Ignored**: Samples with no remaining labels after filtering were dropped. While this improves model stability, it may affect generalization in real-world deployment where unseen labels exist.

## Summary

The XGBoost-based multi-label classifier achieves high precision and recall across a wide spectrum of mutation tags, even in a high-dimensional and partially imbalanced setting. Given its performance and efficiency, it is well-suited for deployment in bioinformatics pipelines or downstream research tools.

**6. Conclusion and Recommendations**

**6.1 Conclusion**

This project successfully implemented a multi-label classification system to identify mutation tags from high-dimensional biological data. Starting from raw .data and .tags files, we executed a full machine learning pipeline involving:

* **Data Cleaning**: Structured raw data into usable format; handled missing values and aligned features with mutation labels.
* **Exploratory Data Analysis**: Assessed data quality, label imbalance, and feature correlation; applied dimensionality reduction using Truncated SVD.
* **Model Selection**: Compared multiple models (Logistic Regression, Random Forest, LightGBM, XGBoost) using One-vs-Rest strategy.
* **Model Analysis**: Final XGBoost model achieved exceptional micro and macro F1 scores (>99%), showing robustness across frequent and infrequent mutation tags.

Through systematic preprocessing and filtering, we reduced noise, increased label clarity, and delivered a highly accurate model for multi-label mutation detection.

**6.2 Recommendations**

Based on the results and observations, we suggest the following next steps:

1. **Deploy the XGBoost Model**  
   Save and serve the trained model as an API or integrate it into a research workflow for real-time mutation prediction.
2. **Handle Rare Tags in Future Iterations**  
   Introduce techniques like synthetic oversampling (e.g., SMOTE) or meta-labeling to address very rare but potentially significant mutation tags.
3. **Add Interpretability Layer**  
   Use tools like **SHAP** or **LIME** to provide transparency into what features influence predictions — crucial for bioinformatics applications.
4. **Continuous Learning and Feedback Loop**  
   Allow new data and labels to be fed back into the pipeline periodically, enabling the model to stay updated with evolving mutation patterns.
5. **Evaluate on External Datasets**  
   To test generalizability, evaluate the model on an independent external dataset or cross-laboratory data if available.

**7. Appendix**

This appendix includes key code blocks and sample outputs that support the technical workflow presented in this project. Each section corresponds to a stage in the machine learning pipeline.

**A.1 Data Cleaning and Preparation**

**Parsing and Saving Cleaned Tags**

with open(tag\_path, "r") as f:

for line in f:

tag = line.strip().strip('%')

tags.append(tag)

cleaned\_tags\_df = pd.DataFrame({'id': list(range(len(tags))), 'tags': tags})

cleaned\_tags\_df.to\_csv(output\_path, index=False)

**Loading and Aligning Feature Data**

X = pd.read\_csv(data\_path, header=None)

tags\_df = pd.read\_csv(tag\_path)

X['id'] = tags\_df['id']

X['tags'] = tags\_df['tags']

**Handling Misaligned Columns**

for chunk in pd.read\_csv(data\_path, header=None, na\_values='?', chunksize=1000):

chunk\_list.append(chunk)

X = pd.concat(chunk\_list, ignore\_index=True)

if X.shape[1] > 5410:

X = X.iloc[:, :5410]

**A.2 Tag Analysis and Label Encoding**

**Splitting and Counting Tags**

X['num\_tags'] = X['tags'].apply(lambda x: len(x.split('\_')) if isinstance(x, str) else 0)

**Multi Label Binarization**

from sklearn.preprocessing import MultiLabelBinarizer

X['tag\_list'] = X['tags'].str.split('\_')

mlb = MultiLabelBinarizer()

Y = mlb.fit\_transform(X['tag\_list'])

tag\_classes = mlb.classes\_

**Saving Cleaned Data**

X.to\_csv(os.path.join(processed\_dir, 'cleaned\_features.csv'), index=False)

np.save(os.path.join(processed\_dir, 'encoded\_labels.npy'), Y)

**A.3 Exploratory Data Analysis (EDA)**

**Dataset Summary**

print(f"📄 Features shape: {X.shape}")

print(f"🏷️ Labels shape: {Y.shape}")

print(f"🔖 Number of tag classes: {len(tag\_classes)}")

**Feature Sparsity and Tag Distribution**

python

CopyEdit

feature\_sparsity = 100.0 \* (X == 0).sum().sum() / (X.shape[0] \* X.shape[1])

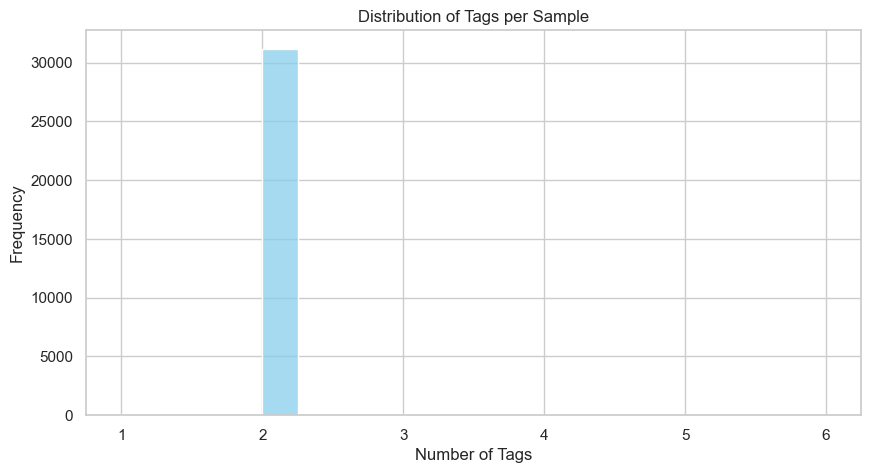
tags\_per\_sample = Y.sum(axis=1)

**Top and Least Frequent Tags**

tag\_frequencies = Y.sum(axis=0)

most\_common\_tags = np.argsort(tag\_frequencies)[::-1][:10]

least\_common\_tags = np.argsort(tag\_frequencies)[:10]



**A.4 Dimensionality Reduction**

**Truncated SVD to Reduce Features**

from sklearn.decomposition import TruncatedSVD

X\_sparse = csr\_matrix(X\_numeric)

svd = TruncatedSVD(n\_components=150, random\_state=42)

X\_reduced = svd.fit\_transform(X\_sparse)

**Output**

Dimensionality reduced. New shape: (31420, 150)

Variance explained: 1.0000

**A.5 Model Training and Evaluation**

**XGBoost Model Setup**

from xgboost import XGBClassifier

xgb\_model = OneVsRestClassifier(XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss'))

xgb\_model.fit(X\_train, Y\_train)

**Evaluation**

from sklearn.metrics import classification\_report

print(classification\_report(Y\_test, Y\_pred\_xgb))

Classification Report for XGBoost:

precision recall f1-score support

0 1.00 1.00 1.00 724

1 0.97 0.98 0.98 122

2 1.00 1.00 1.00 694

3 1.00 1.00 1.00 13

4 1.00 0.88 0.94 17

5 0.98 0.97 0.98 118

6 0.99 0.95 0.97 133

7 0.99 0.98 0.99 111

8 0.99 0.98 0.98 131

9 1.00 1.00 1.00 732

10 0.98 0.94 0.96 126

11 1.00 0.99 1.00 739

12 0.99 0.95 0.97 123

13 1.00 1.00 1.00 774

14 1.00 0.98 0.99 126

15 0.99 0.97 0.98 118

16 1.00 1.00 1.00 739

17 1.00 1.00 1.00 742

micro avg 1.00 0.99 1.00 6282

macro avg 0.99 0.98 0.99 6282

weighted avg 1.00 0.99 1.00 6282

samples avg 1.00 0.99 0.99 6282

**A.6 Model Export**

import joblib

model\_path = os.path.join("../saved\_model", "xgb\_multilabel\_model.pkl")

joblib.dump(xgb\_model, model\_path)