UITS

UNIVERSITY OF INFORMATION TECHNOLOGY AND SCIENCES

Department of Computer Science and Engineering

Bioinformatics and Computational Biology LAB
CSE 430

Submission Date
06/01/2025

Report

Project: Classification of Conditions Using Gene Expression (Blood Cancer Gene)

Submitted to

Jobair Ahammed Tusher
Lecturer
CSE Dept. UITS

Submitted By

Ishraq Uddin Chowdhury
2114951040
Section 8A2

Project Title: Classification of Conditions Using Gene Expression

Abstract

This project explores the classification of biological conditions based on gene expression data. We implemented a machine learning pipeline using the dataset provided, which includes gene expression levels in FPKM format across different conditions. The Random Forest algorithm was utilized to classify conditions with high accuracy. The project aimed to uncover significant genes, analyze their role in classification and visualize the insights gained through statistical and machine learning techniques.

Introduction

Gene expression data is critical in understanding biological functions and conditions. This project leverages gene expression levels under four conditions: DSF, IM, and IM.DSF. The dataset is in FPKM format, normalizing read counts based on transcript length and sequencing depth.

The objectives of this project are:

1. To preprocess and normalize the gene expression data.
2. To identify significant genes for classification using statistical methods.
3. To train a machine learning model to classify conditions.
4. To visualize and interpret the results for biological insights.

Dataset Description

The dataset comprises the following columns:

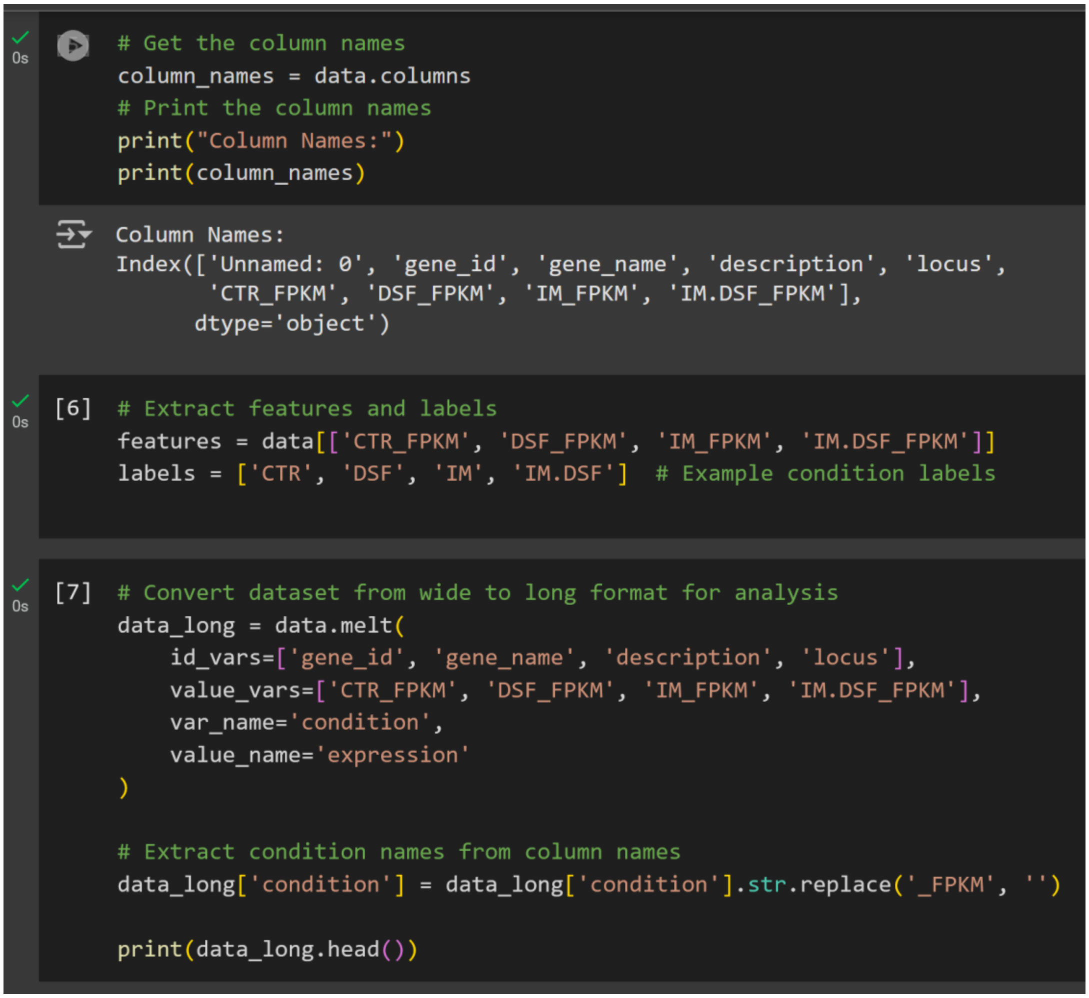
- **gene_id**: Unique identifier for each gene.
- **gene_name**: Name of the gene.
- **description**: Functional description of the gene.
- **locus**: Chromosomal location of the gene.
- **CTR_FPKM, DSF_FPKM, IM_FPKM, IM.DSF_FPKM**: Expression levels of genes under different conditions.

Methods

1. Data Preprocessing

- Normalized the dataset using StandardScaler to standardize expression levels.
- Assigned conditions based on maximum expression across columns using pandas.
- Encoded the conditions into numeric labels for machine learning.

Code:



```
# Get the column names
column_names = data.columns
# Print the column names
print("Column Names:")
print(column_names)
```

```
Column Names:
Index(['Unnamed: 0', 'gene_id', 'gene_name', 'description', 'locus',
       'CTR_FPKM', 'DSF_FPKM', 'IM_FPKM', 'IM.DSF_FPKM'],
      dtype='object')
```

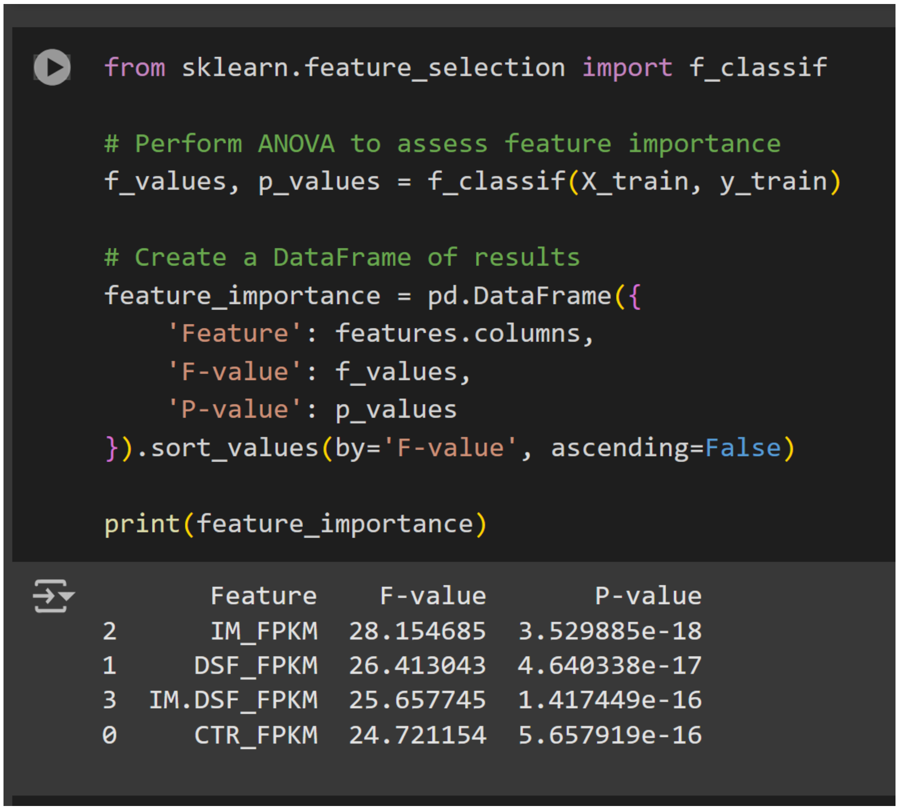
```
# Extract features and labels
features = data[['CTR_FPKM', 'DSF_FPKM', 'IM_FPKM', 'IM.DSF_FPKM']]
labels = ['CTR', 'DSF', 'IM', 'IM.DSF']  # Example condition labels
```

```
# Convert dataset from wide to long format for analysis
data_long = data.melt(
    id_vars=['gene_id', 'gene_name', 'description', 'locus'],
    value_vars=['CTR_FPKM', 'DSF_FPKM', 'IM_FPKM', 'IM.DSF_FPKM'],
    var_name='condition',
    value_name='expression'
)

# Extract condition names from column names
data_long['condition'] = data_long['condition'].str.replace('_FPKM', '')

print(data_long.head())
```

2. Feature Selection

Performed ANOVA to identify significant genes based on their F-values and p-values. Genes with p-values < 0.05 were selected as features for the model.

Code:



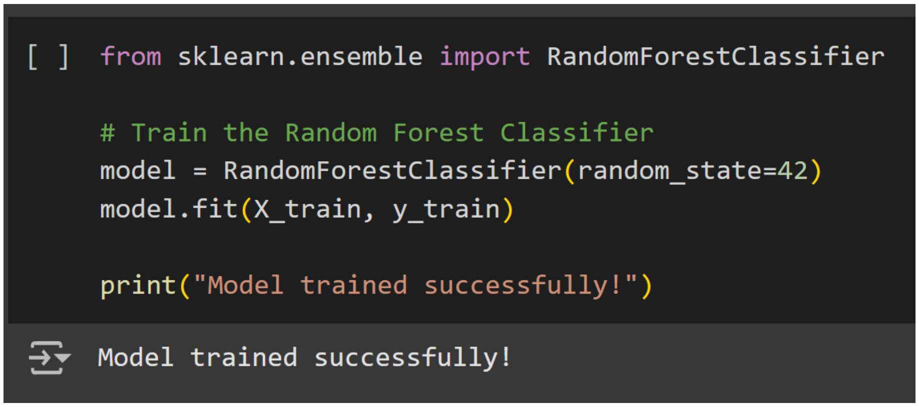
```
from sklearn.feature_selection import f_classif

# Perform ANOVA to assess feature importance
f_values, p_values = f_classif(X_train, y_train)

# Create a DataFrame of results
feature_importance = pd.DataFrame({
    'Feature': features.columns,
    'F-value': f_values,
    'P-value': p_values
}).sort_values(by='F-value', ascending=False)

print(feature_importance)
```

```
       Feature    F-value       P-value
2      IM_FPKM  28.154685  3.529885e-18
1     DSF_FPKM  26.413043  4.640338e-17
3  IM.DSF_FPKM  25.657745  1.417449e-16
0     CTR_FPKM  24.721154  5.657919e-16
```

3. Model Training

Trained a Random Forest Classifier using the significant features identified. This model was chosen for its robustness and ability to handle complex datasets.

Code:



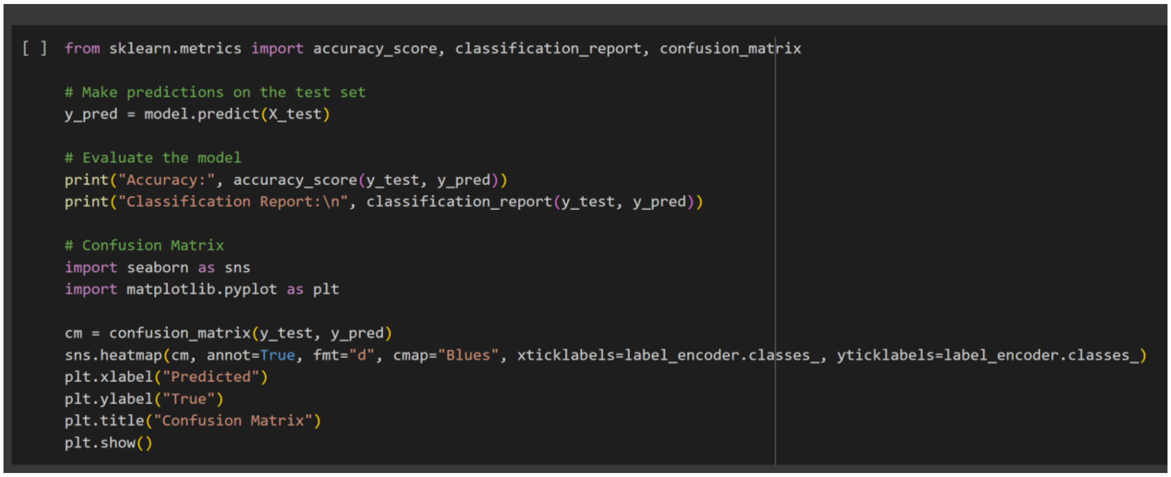
```
[ ]  from sklearn.ensemble import RandomForestClassifier

     # Train the Random Forest Classifier
     model = RandomForestClassifier(random_state=42)
     model.fit(X_train, y_train)

     print("Model trained successfully!")
```

```
Model trained successfully!
```

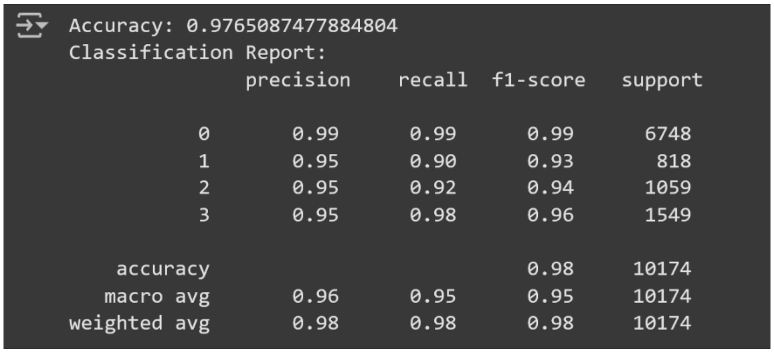
4. Model Evaluation

Evaluated the model using metrics like accuracy, precision, recall, and F1-score. A confusion matrix was plotted to visualize the prediction results.

Code:



```
[ ]  from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

     # Make predictions on the test set
     y_pred = model.predict(X_test)

     # Evaluate the model
     print("Accuracy:", accuracy_score(y_test, y_pred))
     print("Classification Report:\n", classification_report(y_test, y_pred))

     # Confusion Matrix
     import seaborn as sns
     import matplotlib.pyplot as plt

     cm = confusion_matrix(y_test, y_pred)
     sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
     plt.xlabel("Predicted")
     plt.ylabel("True")
     plt.title("Confusion Matrix")
     plt.show()
```



```
Accuracy: 0.9765087477884804
Classification Report:
               precision    recall  f1-score   support

           0       0.99      0.99      0.99      6748
           1       0.95      0.90      0.93       818
           2       0.95      0.92      0.94      1059
           3       0.95      0.98      0.96      1549

    accuracy                           0.98     10174
   macro avg       0.96      0.95      0.95     10174
weighted avg       0.98      0.98      0.98     10174
```

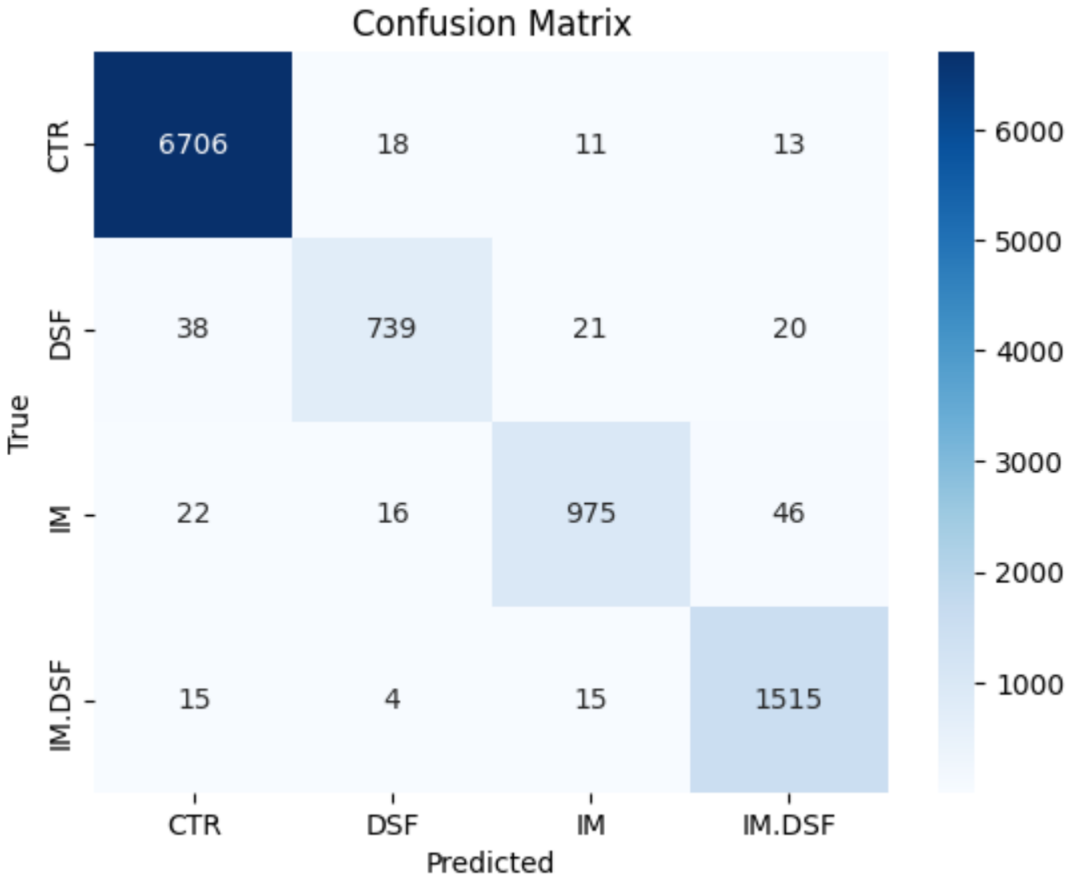
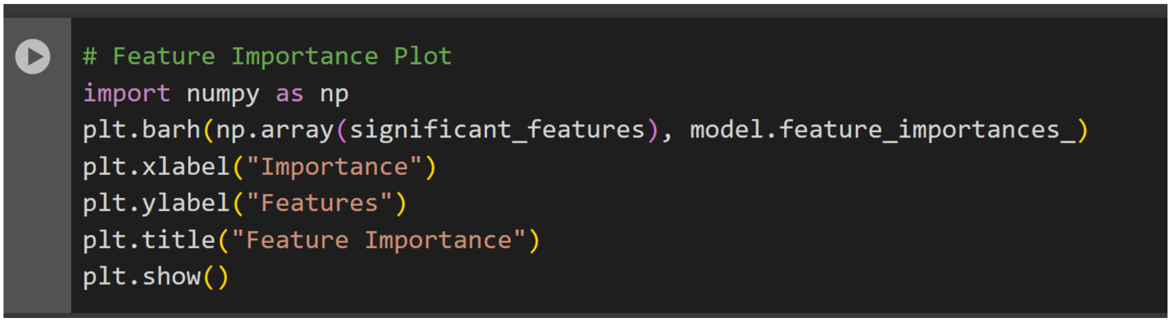


fig1: Confusion Matrix of Classification

5. Visualization

The plotted feature is important to identify the genes contributing most to classification. Visualized gene expression distributions using boxplots.

Code:



```
# Feature Importance Plot
import numpy as np
plt.barh(np.array(significant_features), model.feature_importances_)
plt.xlabel("Importance")
plt.ylabel("Features")
plt.title("Feature Importance")
plt.show()
```

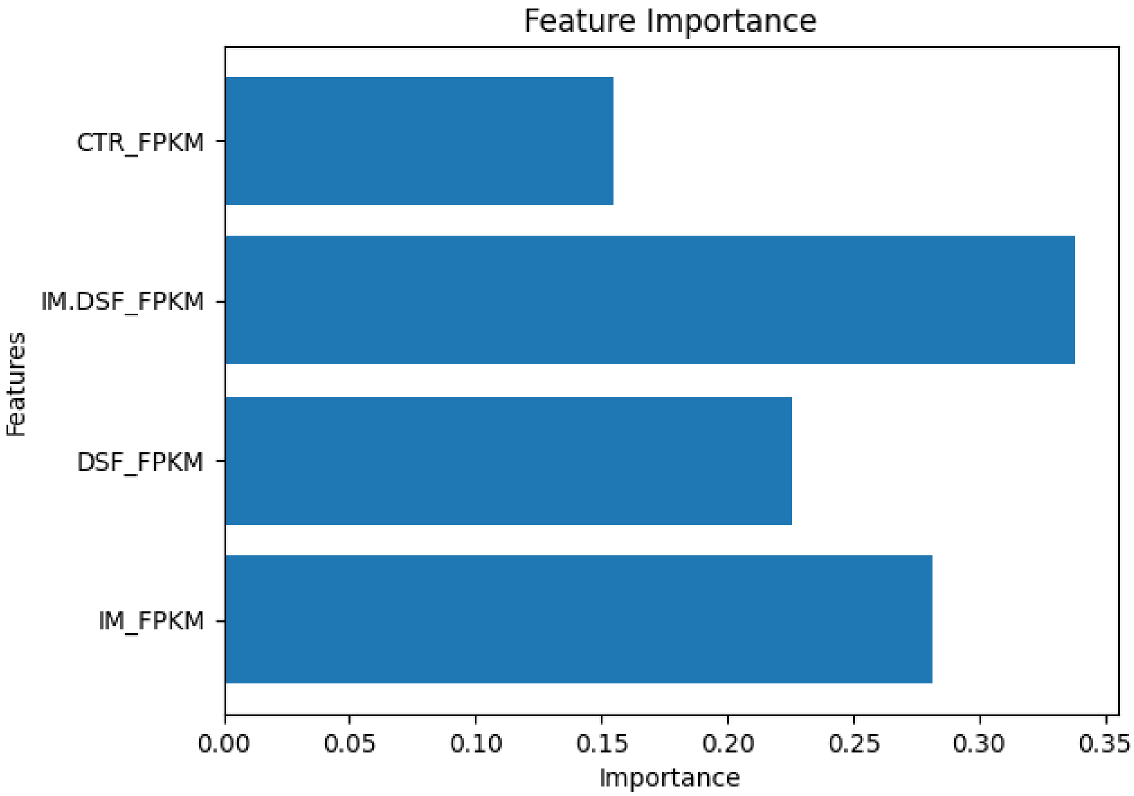
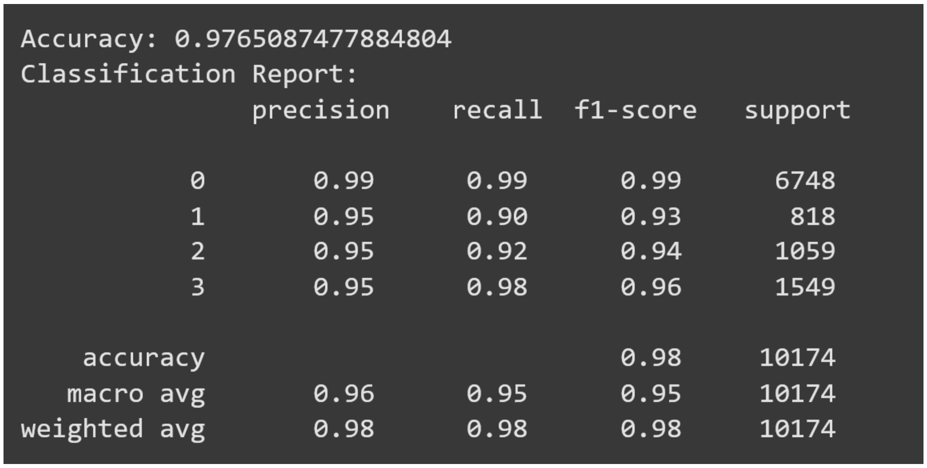


Fig2: Feature Importance

Results



```
Accuracy: 0.9765087477884804
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      6748
           1       0.95      0.90      0.93       818
           2       0.95      0.92      0.94      1059
           3       0.95      0.98      0.96      1549

    accuracy                           0.98     10174
   macro avg       0.96      0.95      0.95     10174
weighted avg       0.98      0.98      0.98     10174
```

- **Model Accuracy**: 97%
- **Significant Genes**: Identified a subset of genes with p-values < 0.05 contributing significantly to condition classification.
- **Confusion Matrix**: The model could distinguish conditions with minimal misclassifications.
- **Feature Importance**: Highlighted key genes that drive classification.

Future Work

- Expand the analysis to include pathway enrichment and functional annotation of significant genes.
- Explore deep learning approaches for improved classification performance.
- Investigate the biological implications of identified significant genes.

Libraries Used

- **pandas**: For data manipulation and cleaning.
- **scikit-learn**: For preprocessing, feature selection, model training, and evaluation.
- **seaborn/matplotlib**: For visualization.

Conclusion

This project successfully applied bioinformatics and machine learning techniques to classify conditions based on gene expression data. The methodology and results showcase the potential of integrating statistical methods and machine learning for biological data analysis.