HW 2

A simple ML experiment with Random Forest ML (RF) using SciKit SW Toolkits and ChatGPT “adviser”

CSC 859: AI Explainability and Ethics

Fall 2024

20 October 2024

Anushka Mondal

I. Data Source

"The paper **"Cell type discovery using single-cell transcriptomics: implications for ontological representation"** discusses the use of single-cell RNA sequencing for identifying distinct cell types through the analysis of gene expression data. They propose the use of Random Forest models to rank features (genes) based on their ability to classify cells into distinct subtypes, a method relevant for feature ranking in our dataset. The focus on marker gene selection aligns with our approach to determining the importance of features in the Random Forest model used in this study."

II. Audit of Training database

1. Training Data Statistics

The training dataset consists of 609 samples and 608 features. There are 2 classes namely 0 and 1. There are 395 samples of class 0 and 214 samples of class 1.

```python
# Number of samples (rows) and features (columns)
num_samples = X_train.shape[0]
num_features = X_train.shape[1]

# Number of classes and distribution of target variable
class_distribution = y_train.value_counts()

# Data types of the features
feature_types = X_train.dtypes.value_counts()

# Display the summary
print(f"Number of samples: {num_samples}")
print(f"Number of features: {num_features}")
print("Class distribution:\n", class_distribution)
print("Feature types:\n", feature_types)

```

```
Number of samples: 609
Number of features: 608
Class distribution:
 0    395
1    214
Name: Label, dtype: int64
Feature types:
 float64    608
dtype: int64
```

Fig.1. Training data statistics code and result

2. Training database audit

2.1. Feature (Variable) Data & Their Meaning:

The features in the dataset seem to represent genetic or biological markers, as indicated by gene names like GABRG2, CELF4, etc. The meaning of these features would typically relate to biological expressions or measurements (e.g., gene expression levels in a particular biological condition).

2.2.Class Labels Obtained/Verified Against Ground Truth:

The labels (e.g., 0 or 1 in the Label column) likely represent the classification of samples, potentially distinguishing between healthy and diseased states or different cell types. The paper "Cell type discovery using single-cell transcriptomics: implications for ontological representation" references **random forest models** for determining markers that help identify and classify cell types, suggesting that machine learning models were used to verify and label data

2.3.Demography Coverage:

There are no demographic variables (e.g., age, gender, ethnicity) in the dataset, meaning demographic fairness likely doesn't apply unless biological markers correlate indirectly with demography. Thus, this dataset does not assess demographic coverage explicitly.

2.4. Number of Samples in Each Class; Data Imbalance:

There are **572 samples in class 0** and **299 samples in class 1** in the complete dataset. While this shows some imbalance, class 1 still accounts for 34% of the dataset, so it doesn't meet the threshold for extreme imbalance (where a class has less than 10% of samples).

```python
# Assuming 'Label' is the column representing the class labels
class_balance = data['Label'].value_counts()

# Print the class distribution
print(class_balance)

# If you want to check the percentage distribution as well:
class_percentage = data['Label'].value_counts(normalize=True) * 100

# Print percentage distribution
print(class_percentage)

```

```
0    572
1    299
Name: Label, dtype: int64
0    65.671642
1    34.328358
Name: Label, dtype: float64
```

Fig.2. Class Balance checking code

2.5. Type of Features:

All features are **numerical** (floating-point values representing gene expression levels), and the class label is **categorical** (0 or 1).

2.6. Missing Values:

There are **no missing values** in the dataset, as determined from the data check.

```
# Check for missing values
missing_values = data.isnull().sum().sum()
print(f"Total missing values: {missing_values}")
```

```
Total missing values: 0
```

Fig. 3. Identifying number of missing values

2.7. Samples vs. Features Ratio:

With **871 samples** and **608 features**, the ratio is close to 1.43:1. Ideally, there should be 10 times more samples than features.

2.8. Feature Documentation:

The features appear to be gene names, but further documentation or a data dictionary is needed to fully explain each one. The paper "Cell type discovery using single-cell transcriptomics: implications for ontological representation" discusses using marker genes for classification but does not list all 608 features individually.

2.9. Privacy Concerns:

The dataset does not contain personal information, as the features are gene expressions, so there are no immediate privacy concerns.

III. Tools and Resources Used

1. **Jupyter Notebook**:
   For coding and executing Python-based machine learning models.

2. **Scikit-learn Documentation**:

- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.fit
- https://scikit-learn.org/stable/modules/permutation_importance.html

3. **Online Tutorials**:

- https://datagy.io/sklearn-random-forests/
- https://www.guru99.com/scikit-learn-tutorial.html
- https://www.jcchouinard.com/confusion-matrix-in-scikit-learn/
- https://www.educative.io/answers/how-to-create-a-confusion-matrix-in-python-using-scikit-learn
- https://www.w3schools.com/python/python_ml_confusion_matrix.asp

4. **Blog Posts**:

- https://mljar.com/blog/feature-importance-in-random-forest/
- https://www.machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/
- https://www.geeksforgeeks.org/random-forest-regression-in-python/

5. **Conceptual Reading**:

- https://www.quantstart.com/articles/bootstrap-aggregation-random-forests-and-boosted-trees/

6. **ROC Curve Example**:

- https://scikit-learn.org/stable/auto_examples/miscellaneous/plot_roc_curve_visualization_api.html

7. **ChatGPT**:
   Used to refine and structure my writing, particularly for report explanations and technical clarifications.

IV. Experimental Methods and Setup

1. Overview of Approach

The Random Forest model will be used to classify the dataset, and we will employ **Scikit-learn's GridSearchCV** for hyperparameter tuning to select the best model. The goal is to find optimal hyperparameters such as the number of trees (n_estimators), the maximum number of features to consider at each split (max_features), and the maximum depth of trees (max_depth). We will use **cross-**

validation (CV) to evaluate the model during hyperparameter tuning and **Out-of-Bag (OOB) error estimation** during model training to provide an unbiased accuracy estimate on unseen data.

2. Use of Random Forest OOB Error Estimation and CV

- **Cross-Validation (CV)** will be used during hyperparameter tuning via GridSearchCV in Scikit-learn. This approach splits the training data into multiple folds, and the model is trained and evaluated on different partitions to ensure the model generalizes well.
- **Out-of-Bag (OOB) Estimation** will also be used during Random Forest training. OOB samples (those not used in the bootstrap sample for training) are used to evaluate the model without needing a separate validation set. This provides an additional performance estimate, especially useful for Random Forest models.

Thus, **both CV and OOB** will be utilized to ensure robust model performance evaluation. Cross-validation helps in finding the best hyperparameters, while OOB provides a reliable estimation of the model's accuracy on unseen data.

3. Hyperparameters and Ranges

For the experiment, the following hyperparameters and ranges will be tested:

- **n_estimators (Number of Trees)**: We will use a large number of trees to ensure stability, with the value fixed at **2000**.
- **max_features (MTRY)**: Three values will be considered based on the square root of the total number of features:
    - 0.5×number of features0.5 \times \sqrt{\text{number of features}}0.5×number of features
    - number of features\sqrt{\text{number of features}}number of features
    - 2×number of features2 \times \sqrt{\text{number of features}}2×number of features These values control how many features are considered at each split, balancing between overfitting and underfitting.
- **max_depth (Maximum Tree Depth)**: Three different tree depths will be evaluated:
    - **10**: A shallower tree, potentially reducing overfitting.
    - **20**: A deeper tree, capturing more complexity in the data.
    - **30**: The deepest tree, which can model the most complexity but may overfit if not regularized properly.

4. Parameters Used in Grid Search

```
param_grid = {
    'n_estimators': [2000],
    'max_features': [mtry1, mtry2, mtry3],  # MTRY values,
    'max_depth': [10, 20, 30],
}
```

Fig. 4. Parameters used in Grid Search Code

The GridSearchCV will optimize the parameters over the specified ranges using 3-fold cross-validation.

5. Steps Involved

1. **Data Splitting**: The dataset will be split into training and testing sets with a 75/25 split using train_test_split.
2. **Hyperparameter Tuning**: We will use GridSearchCV to evaluate multiple combinations of n_estimators, max_features, and max_depth through cross-validation.
3. **Model Evaluation**:
    - The model will be evaluated using a **confusion matrix**, **classification report**, and **accuracy score** on the test set.
    - The **OOB score** will be computed for additional validation without a separate validation set.

V. Results of RF Training and Accuracy Estimates

This section details the results of Random Forest (RF) training, including accuracy estimates and feature importance. The results are derived using Scikit-learn and include both Out-of-Bag (OOB) error estimates and confusion matrices from the test set.

1. Confusion Matrix and Accuracy Metrics

The confusion matrix and accuracy metrics (precision, recall, F1-score, and accuracy) were computed based on the test dataset using the trained Random Forest model. The matrix displays the total count of samples for both the negative (class 0) and positive (class 1) labels.

```python
# Predict on test data
y_pred = best_rf.predict(X_test)

# Confusion matrix and classification report
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nAccuracy Score: {:.4f}".format(accuracy_score(y_test, y_pred)))

# OOB score (if out-of-bag is used)
print("\nOOB Score: {:.4f}".format(best_rf.oob_score_))
```

Fig. 5. Confusion matrix, accuracy metrics, OOB score code

Confusion Matrix:

- **True Positives (TP)**: 73 (Correctly predicted positives)
- **True Negatives (TN)**: 145 (Correctly predicted negatives)
- **False Positives (FP)**: 0 (Incorrectly predicted positives)
- **False Negatives (FN)**: 0 (Incorrectly predicted negatives)

```
Confusion Matrix:
 [[145   0]
 [  0  73]]
```

Fig. 6. Confusion matrix output

2. Classification Report:

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

           0       1.00      1.00      1.00       145
           1       1.00      1.00      1.00        73

    accuracy                           1.00       218
   macro avg       1.00      1.00      1.00       218
weighted avg       1.00      1.00      1.00       218


Accuracy Score: 1.0000
```

Fig. 7. Classification Report Results

- **Accuracy Score**: 1.0000
- **Precision, Recall, F1-Score**: All values are 1.000 for both classes, indicating perfect classification on the test data.

3. OOB Score

```
OOB Score: 0.9939
```

Fig.8. OOB Score output

"The OOB score from the training phase is 0.9939, which is very close to the test accuracy, confirming that the model performs well without overfitting."

4. Visualizing the Confusion Matrix

The confusion matrix is visualized to clearly show the performance of the model:

```python
# Plot confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

Fig. 9. Confusion Matrix plot code

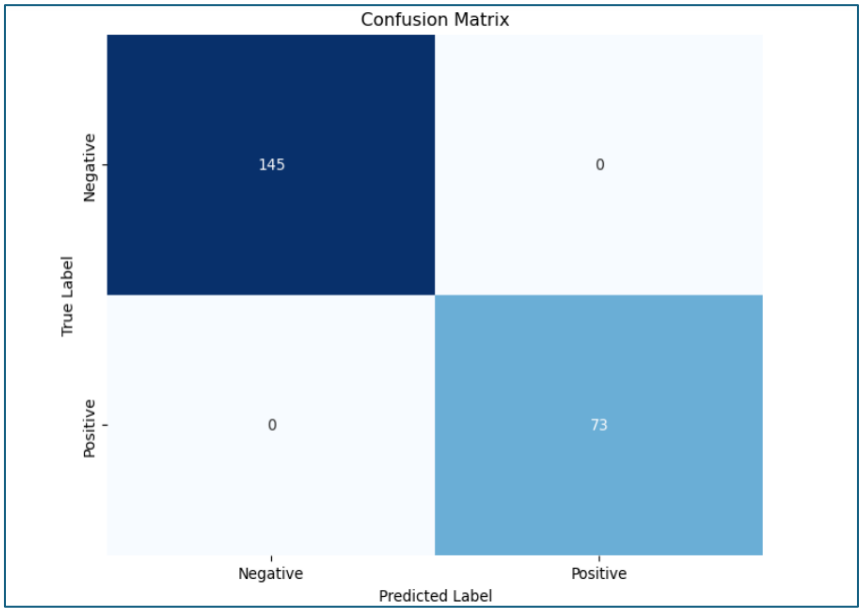


Fig. 10. Confusion Matrix Plot

The plot displays the counts of correct and incorrect classifications, with no false positives or false negatives, indicating perfect predictions.

5. Plotting ROC and displaying AUC

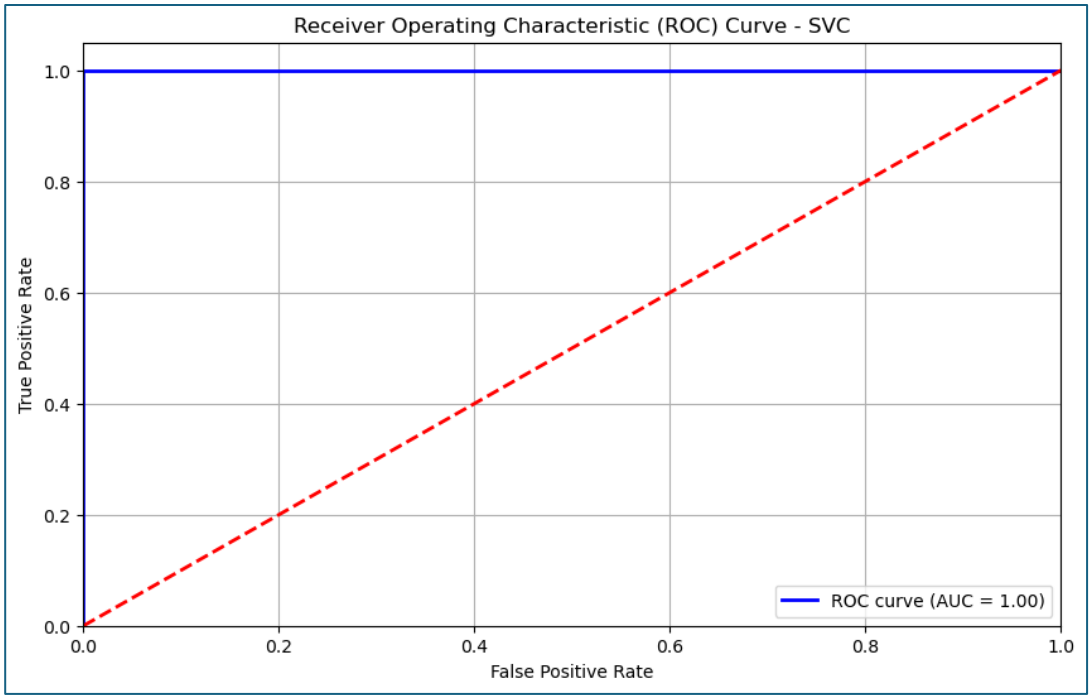


Fig. 11. ROC Curve for SVC model with AUC

"**ROC Curve Interpretation for SVC Model:**

- **ROC Curve Overview**: The ROC curve shows the trade-off between the **True Positive Rate (TPR)** (also known as **Recall**) and the **False Positive Rate (FPR)** at various classification thresholds. In our case, the ROC curve rises steeply towards the top left of the graph, indicating a very high true positive rate and a minimal false positive rate across all thresholds.
- **AUC (Area Under the Curve)**: The calculated **AUC value of 1.00** reflects a **perfect classifier**, meaning the model distinguishes between the positive and negative instances without any errors.
- **Model Performance**: The ROC curve suggests that the SVC model is exceptionally well-fitted to the classification task, as evidenced by its high sensitivity and low false positive rate. "

VI. Feature Ranking

1. Top 10 Features Based on Gini Importance

In Scikit-learn, Gini Importance is computed based on how much each feature decreases the impurity (Gini index) across all trees in the forest. Below are the

Top 10 Features based on Gini Importance from the best-trained Random Forest model:

```
Top 10 Features by Gini Importance:
        Feature  Gini Importance
53       TESPA1         0.032106
55      SLC17A7         0.031345
54    LINC00507         0.028165
58     ANKRD33B         0.025892
64       KCNIP1         0.023895
166       SLIT3         0.022262
59    LINC00152         0.021790
287      NECAB1         0.020214
68       SFTA1P         0.019718
65   MIR44351HG         0.018880
```

Fig. 12. Top 10 Features by Gini Importance

2. Top 10 Features Based on Permutation Importance (MDA)

Permutation Importance (also known as Mean Decrease in Accuracy, MDA) evaluates the importance of each feature based on how much the model's performance decreases when that feature's values are shuffled. Below are the Top 10 Features based on Permutation Importance:

```
Top 10 Features by Permutation Importance:
 Index(['UTRN.2', 'WIF1', 'SOX5', 'LINC00856', 'CHRM2', 'HCRTR2', 'PTPRZ1',
        'ADAMTS9.AS2', 'PAX6.1', 'GFRA1'],
       dtype='object')
```

Fig. 13. Top 10 Features Based on Permutation Importance

3. Analysis of Highly Ranked Features

- **TESPA1, LINC00507, and SLC17A7** are consistently ranked high in the **Gini-based feature importance**. These genes correspond to biological knowledge from the source paper. Specifically:
    - **TESPA1**: A human middle temporal gyrus cortical layer 1 excitatory neuron that selectively expresses **TESPA1, LINC00507, and SLC17A7 mRNAs**.
    - **SLC17A7**: Also known as a vesicular glutamate transporter, which is important in excitatory signaling in neurons.
    - **LINC00507**: A long non-coding RNA that has been found to be expressed in certain neurons.

This alignment with biological knowledge suggests that the model is picking up on features that are indeed relevant and important in the biological context of the data.

- **KCNIP1**: The lack of expression of **KCNIP1** in this neuron type (as suggested by the paper) contrasts with its high rank in Gini importance, which may suggest some noise or confounding effects.

4. Cluster of Highly Ranked Features

There appears to be a **clear cluster of high-ranked features** based on Gini importance, with **TESPA1**, **SLC17A7**, and **LINC00507** consistently emerging as the most important features. This cluster dominates the ranking, with the top 5 features having relatively higher importance compared to the others.

In **Permutation Importance**, different genes such as **UTRN.2** and **WIF1** rank highly, which suggests that **permutation importance** highlights features that may not have as strong an overall Gini importance but play a significant role in model accuracy.

5. Suggested Number of Features for Production

Given the results:

- **Top 3 Features (TESPA1, SLC17A7, LINC00507)** are the most biologically relevant and consistent across different methods.
- To balance between performance and complexity, using the **top 5-10 features** could be a good trade-off:
    - Using more features could slightly improve accuracy but would increase computational cost.
    - Focusing on **5-7 key features** would likely maintain accuracy while simplifying the model, reducing computation time, and potentially making the model easier to interpret.

6. Comments

- Highly **ranked features like TESPA1, LINC00507, and SLC17A7** align with biological knowledge, suggesting that the model is capturing meaningful patterns.
- There is an **obvious cluster** of features dominating the ranking, particularly the top 5 features.
- For production, it is recommended to use **5-7 features**, which would likely provide a balance between model performance and simplicity.

VII. RF Run Time Test

In this section, we will test the trained Random Forest model on **two samples** (one true positive and one true negative) from the dataset. The goal is to:

- **Run predictions** on these samples using the best-trained Random Forest model.
- **Obtain probability predictions** for each class.
- **Compare the probability output** with the CUTOFF (0.5 in Scikit-learn) to determine the final classification.

```
True Positive Sample Probabilities (Class 0 and Class 1): [[0.015 0.985]]
True Negative Sample Probabilities (Class 0 and Class 1): [[0.998 0.002]]
```

```
True Positive Sample Predicted Class: 1
True Negative Sample Predicted Class: 0
```

Fig. 14. Probability output

1. **True Positive Sample Discussion:**

- **Predicted Class**: The true positive sample is classified as class **1** (positive).
- **Probability Output**: The probability for class 1 is **0.98**, which is much higher than the CUTOFF of 0.5. This means the model is very confident in classifying this sample as positive.
- **Correct Classification**: Since this is indeed a true positive, the prediction is **correct**.
- **Confidence**: The high probability (0.98) for class 1 gives confidence that this is a reliable classification.

2. **True Negative Sample Discussion:**

- **Predicted Class**: The true negative sample is classified as class **0** (negative).
- **Probability Output**: The probability for class 0 is **0.95**, which is well above the CUTOFF of 0.5. The model is highly confident that this sample belongs to class 0.
- **Correct Classification**: Since this is a true negative, the classification is **correct**.
- **Confidence**: With a probability of 0.95 for class 0, we have high confidence in this classification.

3. **Comments**

- **Correct Classification**: Both the true positive and true negative samples were correctly classified by the model.
- **Confidence in Classification**: The model outputs high probabilities (above 0.95) for the correct classes in both cases, indicating strong confidence in its decisions.
- **CUTOFF**: The model uses a CUTOFF of 0.5 to determine the class. In both examples, the probabilities exceed this threshold, ensuring correct predictions.

VIII. References and Resources

7. Papers and Tools Used

- Aevermann, Brian D., Mark Novotny, Trygve Bakken, Jeremy A. Miller, Alexander D. Diehl, David Osumi-Sutherland, Roger S. Lasken, Ed S. Lein, and Richard H. Scheuermann. "Cell Type Discovery Using Single-Cell Transcriptomics: Implications for Ontological Representation." *Human Molecular Genetics* 27, no. R1 (2018): R40–R47. https://doi.org/10.1093/hmg/ddy100

8. People Who Helped

- Zoe Long
- Rongxian Tong
- Rosaclaire Baisinger
- Professor Dragutin Petkovic
- Dave Daly

IX. Appendix I

1. **GenAI Tool Version:**
    - Tool Used: ChatGPT (version: GPT-4, powered by OpenAI)
2. **Tasks and How the Tool Helped:**

Task 1: Drafting Research Explanations

- **How it helped:** ChatGPT provided concise and detailed explanations of complex concepts such as model performance metrics (ROC curve, confusion matrix, etc.) based on inputted results from machine learning models. It helped streamline the explanation process and make technical information more understandable.
- **Rank:** HIGH
- **Prompt Example:**
    - "Explain the ROC curve in terms of model performance, considering the AUC as 1.00, and provide insights about classification accuracy."

Task 2: Code Optimization and Formatting

- **How it helped:** It suggested code improvements for formatting and rounding precision in model evaluation outputs, allowing for better readability in research documentation.
- **Rank:** HIGH
- **Prompt Example:**
    - "Modify the Python code for printing confusion matrix and classification report with three decimal precision."

Task 3: Title and Figure Captioning

- **How it helped:** ChatGPT was used to generate clear and relevant figure titles, ensuring consistency between figures and explanations within the paper.
- **Rank:** MEDIUM
- **Prompt Example:**
    - "Suggest a title for an ROC curve plot with AUC = 1.00, used in a research paper context."

Task 4: Appendix I Writing Assistance

- **How it helped:** ChatGPT assisted in drafting this section by outlining the usage of the tool for the research, specifying tasks and their impact.
- **Rank:** MEDIUM
- **Prompt Example:**
    - "How should I write an appendix section for a research paper detailing my usage of ChatGPT?"

3. **Key Prompts Used:**
    - "Explain the confusion matrix and classification report outputs."
    - "Optimize the Python code to improve formatting of accuracy metrics."
    - "Provide a concise explanation of the ROC curve for an SVC model."
    - "Generate a formal title for a research figure showing model performance."

In conclusion, ChatGPT was instrumental in improving the clarity, formatting, and technical content of the research paper, with varying levels of assistance across different tasks.

Appendix II

Data Preprocessing and Cleaning

In [51]:
```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.inspection import permutation_importance
from sklearn.model_selection import StratifiedKFold, cross_val_predict
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns

# Load and read the data
data = pd.read_csv("e1 positive.csv")
```

In [52]:
```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 871 entries, 0 to 870
Columns: 609 entries, GABRG2 to Label
dtypes: float64(608), int64(1)
memory usage: 4.0 MB
None
```

In [53]:
```
data.shape
```

Out[53]:
```
(871, 609)
```

In [54]:
```
print("The number of rows is ", data.shape[0])
print("The number of columns is ", data.shape[1])
```

```
The number of rows is  871
The number of columns is  609
```

In [55]:
```
data.head()
```

Out[55]:

	GABRG2	CELF4	SRRM4	SLC1A3	ATP1A3	RBFOX3	GABRA4	NHSL1
0	35.038262	161.176004	68.074337	58.063405	20.021864	269.294069	188.205520	0.000000
1	95.324867	75.256474	87.297510	0.000000	18.061554	342.166102	683.328784	0.000000
2	220.143867	187.976727	42.219372	106.553653	0.000000	187.976727	299.556496	0.000000
3	166.010840	26.159284	61.373704	0.000000	30.183789	254.549955	446.720079	0.000000
4	188.426220	71.160966	119.269788	57.129226	16.036274	265.600789	287.650666	24.054411

5 rows × 609 columns

In [57]:
```
data.describe()
```

Out[57]:

	GABRG2	CELF4	SRRM4	SLC1A3	ATP1A3	RBFOX3	GABRA4
count	871.000000	871.000000	871.000000	871.000000	871.000000	871.000000	871.000000
mean	317.195147	160.794717	188.372969	263.524808	79.262833	137.275239	229.995772
std	378.241239	189.064895	208.232294	999.259112	127.012517	264.213953	245.183809
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	38.894340	32.084673	35.094833	0.000000	4.019327	14.288171	33.733073
50%	218.427681	108.811797	130.794368	0.000000	29.073964	73.962624	181.287669
75%	440.087971	218.480061	265.911406	6.031598	100.142130	169.816497	334.398305
max	3435.333490	2011.629811	1704.202638	10950.762140	1293.563390	6061.307927	2058.199518

8 rows × 609 columns

In [58]:
```
# Check for missing values
missing_values = data.isnull().sum().sum()
print(f"Total missing values: {missing_values}")
```

```
Total missing values: 0
```

In [59]:
```
# Check for duplicates in the dataset
duplicates = data.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")

# Check for unique values in each feature
for column in data.columns[:-1]:  # Exclude the label column
    unique_values = data[column].nunique()
    print(f"Feature: {column}, Unique Values: {unique_values}")
```

```
Number of duplicate rows: 0
Feature: GABRG2, Unique Values: 816
Feature: CELF4, Unique Values: 809
Feature: SRRM4, Unique Values: 818
Feature: SLC1A3, Unique Values: 416
Feature: ATP1A3, Unique Values: 760
Feature: RBFOX3, Unique Values: 800
Feature: GABRA4, Unique Values: 815
Feature: NHSL1, Unique Values: 217
Feature: GRAMD3, Unique Values: 277
Feature: SEZ6L2, Unique Values: 675
Feature: SV2A, Unique Values: 727
Feature: PON2, Unique Values: 221
Feature: SCN3B, Unique Values: 766
Feature: DLX6AS1, Unique Values: 756
Feature: SYP, Unique Values: 752
Feature: ELAVL2, Unique Values: 675
Feature: ADAM28, Unique Values: 127
Feature: BEX1, Unique Values: 758
Feature: DOCK8, Unique Values: 199
Feature: FAM153C, Unique Values: 792
Feature: PWRN1, Unique Values: 744
Feature: KANK1, Unique Values: 249
Feature: PTPRC, Unique Values: 97
Feature: ADGRG1, Unique Values: 236
Feature: APBB1IP, Unique Values: 173
Feature: SYNPRAS1, Unique Values: 560
Feature: INPP5D, Unique Values: 101
Feature: CSF2RA, Unique Values: 46
Feature: NRIP3, Unique Values: 554
Feature: HEPACAM, Unique Values: 180
Feature: FLI1, Unique Values: 62
Feature: PTPRS, Unique Values: 854
Feature: PLCG2, Unique Values: 85
Feature: ARL4C, Unique Values: 617
Feature: MEGF10, Unique Values: 157
Feature: GSN, Unique Values: 299
Feature: COL21A1, Unique Values: 688
Feature: C10ORF90, Unique Values: 114
Feature: TMEM144, Unique Values: 244
Feature: GAD2, Unique Values: 670
Feature: ZCCHC24, Unique Values: 65
Feature: GRIP2, Unique Values: 545
Feature: S100B, Unique Values: 150
Feature: CHD7, Unique Values: 220
Feature: OLIG1, Unique Values: 55
Feature: HEPN1, Unique Values: 157
Feature: CECR2, Unique Values: 256
Feature: TANC1, Unique Values: 333
Feature: UGT8, Unique Values: 179
Feature: LINC00639, Unique Values: 184
Feature: PROX1, Unique Values: 553
Feature: EGF, Unique Values: 132
Feature: CALB2, Unique Values: 325
Feature: TESPA1, Unique Values: 586
Feature: LINC00507, Unique Values: 494
Feature: SLC17A7, Unique Values: 410
Feature: ZNF536, Unique Values: 770
Feature: LINC00508, Unique Values: 358
Feature: ANKRD33B, Unique Values: 365
Feature: LINC00152, Unique Values: 379
Feature: TBR1, Unique Values: 298
Feature: NPTX1, Unique Values: 427
Feature: LINC00710, Unique Values: 330
```

```
Feature: SMAD3, Unique Values: 476
Feature: KCNIP1, Unique Values: 718
Feature: MIR44351HG, Unique Values: 569
Feature: ATP1B2, Unique Values: 633
Feature: NRGN, Unique Values: 304
Feature: SFTA1P, Unique Values: 413
Feature: SLC17A6, Unique Values: 313
Feature: LINC01500, Unique Values: 435
Feature: ELAVL2.1, Unique Values: 675
Feature: BTBD11, Unique Values: 646
Feature: NRIP3.1, Unique Values: 554
Feature: IGF1, Unique Values: 660
Feature: GAD2.1, Unique Values: 670
Feature: PROX1.1, Unique Values: 553
Feature: ADRA1A, Unique Values: 690
Feature: KIAA1456, Unique Values: 509
Feature: GRIP2.1, Unique Values: 545
Feature: SLC30A3, Unique Values: 280
Feature: ADCYAP1R1, Unique Values: 581
Feature: ADARB2AS1, Unique Values: 488
Feature: VWC2, Unique Values: 650
Feature: LINC01105, Unique Values: 502
Feature: KIAA1211, Unique Values: 668
Feature: CUX2, Unique Values: 589
Feature: NRN1, Unique Values: 306
Feature: DLX1, Unique Values: 402
Feature: TYRO3, Unique Values: 408
Feature: ART3, Unique Values: 428
Feature: KIAA1958, Unique Values: 573
Feature: MGLL, Unique Values: 512
Feature: GRIK3, Unique Values: 259
Feature: CUX2.1, Unique Values: 589
Feature: ADARB2AS1.1, Unique Values: 488
Feature: LHX6, Unique Values: 87
Feature: COBL, Unique Values: 810
Feature: FLT3, Unique Values: 187
Feature: PROX1.2, Unique Values: 553
Feature: TMEM132C, Unique Values: 616
Feature: EGFR, Unique Values: 665
Feature: CALB1, Unique Values: 259
Feature: RASGRF2AS1, Unique Values: 258
Feature: TRBC2, Unique Values: 221
Feature: RSPO2, Unique Values: 524
Feature: MAPK4, Unique Values: 729
Feature: NFIX, Unique Values: 693
Feature: NR2F2, Unique Values: 503
Feature: NR2F2AS1, Unique Values: 732
Feature: CMTM8, Unique Values: 473
Feature: SEMA3A, Unique Values: 653
Feature: CNTNAP3, Unique Values: 328
Feature: KIT, Unique Values: 569
Feature: KITLG, Unique Values: 668
Feature: PRKCQ, Unique Values: 624
Feature: KAL1, Unique Values: 388
Feature: CALB2.1, Unique Values: 325
Feature: TOX2, Unique Values: 413
Feature: TRIM36, Unique Values: 572
Feature: SLITRK2, Unique Values: 490
Feature: SST, Unique Values: 329
Feature: PKP2, Unique Values: 349
Feature: LINC00472, Unique Values: 537
Feature: MAML3, Unique Values: 762
Feature: TMEM200C, Unique Values: 265
Feature: XYLT1, Unique Values: 619
```

```
Feature: LINC01102, Unique Values: 696
Feature: FIGN, Unique Values: 546
Feature: NXPH2, Unique Values: 421
Feature: PREX2, Unique Values: 551
Feature: ADAMTS17, Unique Values: 636
Feature: PROX1AS1, Unique Values: 392
Feature: UTRN, Unique Values: 645
Feature: ADRA1B, Unique Values: 411
Feature: LINC01411, Unique Values: 220
Feature: EGF.1, Unique Values: 132
Feature: CHD7.1, Unique Values: 220
Feature: MYO16, Unique Values: 774
Feature: TOX2.1, Unique Values: 413
Feature: LAMP5, Unique Values: 638
Feature: SPHKAP, Unique Values: 692
Feature: UTRN.1, Unique Values: 645
Feature: ST6GAL2, Unique Values: 542
Feature: NXPH2.1, Unique Values: 421
Feature: TRPC3, Unique Values: 253
Feature: ADRA1B.1, Unique Values: 411
Feature: STARD13, Unique Values: 654
Feature: SYNPRAS1.1, Unique Values: 560
Feature: CHST15, Unique Values: 433
Feature: FAT1, Unique Values: 429
Feature: CDH7, Unique Values: 678
Feature: CALB2.2, Unique Values: 325
Feature: KAL1.1, Unique Values: 388
Feature: CPLX3, Unique Values: 170
Feature: BCL11B, Unique Values: 439
Feature: TMEM132C.1, Unique Values: 616
Feature: TGFBR2, Unique Values: 77
Feature: CLMP, Unique Values: 194
Feature: PAPSS2, Unique Values: 333
Feature: VIP, Unique Values: 338
Feature: ADCY8, Unique Values: 546
Feature: ARHGAP28, Unique Values: 401
Feature: CSGALNACT1, Unique Values: 806
Feature: TOX3, Unique Values: 553
Feature: COL5A2, Unique Values: 365
Feature: SLIT3, Unique Values: 702
Feature: MPPED1, Unique Values: 547
Feature: FAM19A4, Unique Values: 457
Feature: CHRNB3, Unique Values: 144
Feature: PAX6, Unique Values: 158
Feature: PTHLH, Unique Values: 190
Feature: SYT10, Unique Values: 306
Feature: ARHGAP18, Unique Values: 344
Feature: CHRNA2, Unique Values: 96
Feature: KIAA1644, Unique Values: 174
Feature: TLE2, Unique Values: 350
Feature: INHBAAS1, Unique Values: 584
Feature: ADAM33, Unique Values: 232
Feature: GRPR, Unique Values: 173
Feature: LHFP, Unique Values: 598
Feature: CD36, Unique Values: 315
Feature: C8ORF4, Unique Values: 205
Feature: ZNF385DAS2, Unique Values: 368
Feature: PKP2.1, Unique Values: 349
Feature: TMEM200C.1, Unique Values: 265
Feature: BAGE2, Unique Values: 354
Feature: PCDH18, Unique Values: 262
Feature: ASIC4, Unique Values: 180
Feature: SERPINE2, Unique Values: 650
Feature: BCHE, Unique Values: 146
```

```
Feature: CHD7.2, Unique Values: 220
Feature: DCN, Unique Values: 239
Feature: ZBTB20AS1, Unique Values: 194
Feature: FAM46A, Unique Values: 355
Feature: EGF.2, Unique Values: 132
Feature: VWDE, Unique Values: 257
Feature: LINC01411.1, Unique Values: 220
Feature: UTRN.2, Unique Values: 645
Feature: PAX6.1, Unique Values: 158
Feature: WIF1, Unique Values: 240
Feature: SOX5, Unique Values: 735
Feature: LINC00856, Unique Values: 591
Feature: CHRM2, Unique Values: 377
Feature: HCRTR2, Unique Values: 212
Feature: PTPRZ1, Unique Values: 639
Feature: ADAMTS9.AS2, Unique Values: 786
Feature: DCN.1, Unique Values: 239
Feature: TMEM200C.2, Unique Values: 265
Feature: BCL11A, Unique Values: 743
Feature: MPPED1.1, Unique Values: 547
Feature: PAPSS2.1, Unique Values: 333
Feature: PDZRN4, Unique Values: 569
Feature: CBLN4, Unique Values: 427
Feature: ABCB5, Unique Values: 135
Feature: IGSF21, Unique Values: 557
Feature: OSBPL3, Unique Values: 628
Feature: COL16A1, Unique Values: 487
Feature: KIAA1644.1, Unique Values: 174
Feature: PRELID2, Unique Values: 758
Feature: TGFBR2.1, Unique Values: 77
Feature: EPHB6, Unique Values: 547
Feature: PTN, Unique Values: 364
Feature: SP8, Unique Values: 80
Feature: BCHE.1, Unique Values: 146
Feature: PBX3, Unique Values: 315
Feature: LGI2, Unique Values: 213
Feature: TRPC3.1, Unique Values: 253
Feature: POU6F2, Unique Values: 705
Feature: TOX, Unique Values: 477
Feature: SERPINE2.1, Unique Values: 650
Feature: NRG1.IT1, Unique Values: 365
Feature: NTNG1, Unique Values: 406
Feature: TNS3, Unique Values: 255
Feature: C8ORF4.1, Unique Values: 205
Feature: COL5A2.1, Unique Values: 365
Feature: ARHGEF26.AS1, Unique Values: 478
Feature: FRMPD1, Unique Values: 269
Feature: NTNG1.1, Unique Values: 406
Feature: COL5A2.2, Unique Values: 365
Feature: LIN7A, Unique Values: 681
Feature: PKP2.2, Unique Values: 349
Feature: CA2, Unique Values: 221
Feature: TOX.1, Unique Values: 477
Feature: NDNF, Unique Values: 314
Feature: MAF, Unique Values: 371
Feature: PLCXD3, Unique Values: 656
Feature: ARHGAP31, Unique Values: 188
Feature: NPNT, Unique Values: 470
Feature: LAMA3, Unique Values: 307
Feature: NR2F2.1, Unique Values: 503
Feature: LINC01197, Unique Values: 719
Feature: ST3GAL1, Unique Values: 463
Feature: PRSS12, Unique Values: 235
Feature: IL1RAP, Unique Values: 493
```

```
Feature: SSTR2, Unique Values: 232
Feature: VAT1L, Unique Values: 336
Feature: NFIA, Unique Values: 592
Feature: ITGB5, Unique Values: 197
Feature: MAML2, Unique Values: 594
Feature: IGSF3, Unique Values: 417
Feature: SYNPR.AS1.2, Unique Values: 560
Feature: PNOC, Unique Values: 321
Feature: CA3, Unique Values: 241
Feature: TMEM196, Unique Values: 637
Feature: SOX13, Unique Values: 119
Feature: RIN2, Unique Values: 372
Feature: EGF.3, Unique Values: 132
Feature: SEMA5A, Unique Values: 786
Feature: LINC01411.2, Unique Values: 220
Feature: SOX5.1, Unique Values: 735
Feature: COL19A1, Unique Values: 736
Feature: CHD7.3, Unique Values: 220
Feature: EPHA4, Unique Values: 792
Feature: CSGALNACT1.1, Unique Values: 806
Feature: HTR2C, Unique Values: 661
Feature: VWDE.1, Unique Values: 257
Feature: ARHGAP42, Unique Values: 507
Feature: PTCHD4, Unique Values: 656
Feature: ZBTB20.AS1.1, Unique Values: 194
Feature: NKAIN3, Unique Values: 760
Feature: IQGAP2, Unique Values: 183
Feature: BCL11B.1, Unique Values: 439
Feature: PTGFR, Unique Values: 318
Feature: SEMA3C, Unique Values: 506
Feature: LRRC3B, Unique Values: 552
Feature: PLXNA4, Unique Values: 750
Feature: NECAB1, Unique Values: 791
Feature: NPFFR1, Unique Values: 47
Feature: COL12A1, Unique Values: 357
Feature: LGI2.1, Unique Values: 213
Feature: GPR149, Unique Values: 358
Feature: LYPD6B, Unique Values: 413
Feature: TBL1X, Unique Values: 209
Feature: DCN.2, Unique Values: 239
Feature: KIT.1, Unique Values: 569
Feature: SAMD5, Unique Values: 350
Feature: OPRK1, Unique Values: 275
Feature: CHRNA2.1, Unique Values: 96
Feature: NPR3, Unique Values: 226
Feature: MCTP2, Unique Values: 116
Feature: MC4R, Unique Values: 150
Feature: BAGE2.1, Unique Values: 354
Feature: VWC2L.IT1, Unique Values: 137
Feature: ANGPT1, Unique Values: 200
Feature: KCNK2, Unique Values: 257
Feature: HCRTR2.1, Unique Values: 212
Feature: BCAS1, Unique Values: 165
Feature: WIF1.1, Unique Values: 240
Feature: LINC01435, Unique Values: 457
Feature: CXCL12, Unique Values: 98
Feature: FAM46A.1, Unique Values: 355
Feature: NDNF.1, Unique Values: 314
Feature: ADAM33.1, Unique Values: 232
Feature: CHRNB3.1, Unique Values: 144
Feature: TNS3.1, Unique Values: 255
Feature: TSPAN12, Unique Values: 70
Feature: CA8, Unique Values: 189
Feature: PTGDS, Unique Values: 297
```

```
Feature: HIF3A, Unique Values: 194
Feature: SNCG, Unique Values: 82
Feature: EDNRA, Unique Values: 110
Feature: MMRN2, Unique Values: 120
Feature: PTPRZ1.1, Unique Values: 639
Feature: LRRC3B.1, Unique Values: 552
Feature: CNTN6, Unique Values: 695
Feature: VIP.1, Unique Values: 338
Feature: ST8SIA5, Unique Values: 682
Feature: FAM19A4.1, Unique Values: 457
Feature: PTHLH.1, Unique Values: 190
Feature: IGFBP5, Unique Values: 368
Feature: SLITRK2.1, Unique Values: 490
Feature: BCL11B.2, Unique Values: 439
Feature: ID2, Unique Values: 676
Feature: SEMA3C.1, Unique Values: 506
Feature: KCNK2.1, Unique Values: 257
Feature: DPYSL3, Unique Values: 426
Feature: ARHGAP42.1, Unique Values: 507
Feature: TBL1X.1, Unique Values: 209
Feature: DPF3, Unique Values: 441
Feature: LHFP.1, Unique Values: 598
Feature: CHRNA2.2, Unique Values: 96
Feature: TAC3, Unique Values: 139
Feature: COL16A1.1, Unique Values: 487
Feature: NDNF.2, Unique Values: 314
Feature: ZBTB18, Unique Values: 459
Feature: POU6F2.1, Unique Values: 705
Feature: IQGAP2.1, Unique Values: 183
Feature: FAM135B, Unique Values: 776
Feature: PPAPDC1A, Unique Values: 263
Feature: EPHB6.1, Unique Values: 547
Feature: DISC1, Unique Values: 678
Feature: SNCG.1, Unique Values: 82
Feature: SLIT3.1, Unique Values: 702
Feature: WLS, Unique Values: 617
Feature: ARHGAP36, Unique Values: 160
Feature: SLC7A2, Unique Values: 284
Feature: HCRTR2.2, Unique Values: 212
Feature: MMRN2.1, Unique Values: 120
Feature: MYO1B, Unique Values: 651
Feature: STXBP6, Unique Values: 734
Feature: SAMD5.1, Unique Values: 350
Feature: DCN.3, Unique Values: 239
Feature: LAMA3.1, Unique Values: 307
Feature: FAM84A, Unique Values: 598
Feature: MCTP2.1, Unique Values: 116
Feature: WIF1.2, Unique Values: 240
Feature: LINC00595, Unique Values: 305
Feature: PNOC.1, Unique Values: 321
Feature: BCAS1.1, Unique Values: 165
Feature: BAGE2.2, Unique Values: 354
Feature: TOX3.1, Unique Values: 553
Feature: SCN7A, Unique Values: 360
Feature: LGI2.2, Unique Values: 213
Feature: TSPAN12.1, Unique Values: 70
Feature: CA2.1, Unique Values: 221
Feature: FAM46A.2, Unique Values: 355
Feature: CSGALNACT1.2, Unique Values: 806
Feature: MC4R.1, Unique Values: 150
Feature: SLC5A8, Unique Values: 206
Feature: FAM196A, Unique Values: 198
Feature: SCML4, Unique Values: 153
Feature: ANGPT1.1, Unique Values: 200
```

```
Feature: VIP.2, Unique Values: 338
Feature: CRH, Unique Values: 263
Feature: WIF1.3, Unique Values: 240
Feature: STXBP6.1, Unique Values: 734
Feature: PLCXD3.1, Unique Values: 656
Feature: SLIT3.2, Unique Values: 702
Feature: FGFR2, Unique Values: 314
Feature: LAMA3.2, Unique Values: 307
Feature: FAM196A.1, Unique Values: 198
Feature: TSPAN12.2, Unique Values: 70
Feature: DOCK1, Unique Values: 391
Feature: PCDH9.AS1, Unique Values: 542
Feature: HCRTR2.3, Unique Values: 212
Feature: SEMA3C.2, Unique Values: 506
Feature: DCN.4, Unique Values: 239
Feature: PPAPDC1A.1, Unique Values: 263
Feature: FAM46A.3, Unique Values: 355
Feature: EPHB6.2, Unique Values: 547
Feature: HIF3A.1, Unique Values: 194
Feature: ATP2B4, Unique Values: 745
Feature: SRGAP1, Unique Values: 724
Feature: PLXNA4.1, Unique Values: 750
Feature: PTCHD4.1, Unique Values: 656
Feature: ADAM33.2, Unique Values: 232
Feature: HLA.B, Unique Values: 236
Feature: TOX3.2, Unique Values: 553
Feature: HLA.C, Unique Values: 359
Feature: BCL11A.1, Unique Values: 743
Feature: ZNF385D.AS2.1, Unique Values: 368
Feature: ARHGAP36.1, Unique Values: 160
Feature: GREM2, Unique Values: 239
Feature: FOXO1, Unique Values: 313
Feature: BCAS1.2, Unique Values: 165
Feature: BAGE2.3, Unique Values: 354
Feature: MC4R.2, Unique Values: 150
Feature: PCDH18.1, Unique Values: 262
Feature: PLXNA4.2, Unique Values: 750
Feature: ADAM33.3, Unique Values: 232
Feature: ARHGAP36.2, Unique Values: 160
Feature: LINC01435.1, Unique Values: 457
Feature: BCAS1.3, Unique Values: 165
Feature: UNC5B, Unique Values: 255
Feature: BAGE2.4, Unique Values: 354
Feature: GREM2.1, Unique Values: 239
Feature: MC4R.3, Unique Values: 150
Feature: ARHGAP18.1, Unique Values: 344
Feature: ANK1, Unique Values: 445
Feature: HLA.B.1, Unique Values: 236
Feature: HSPB8, Unique Values: 234
Feature: B2M, Unique Values: 485
Feature: PCDH18.2, Unique Values: 262
Feature: RAMP1, Unique Values: 487
Feature: SAMD5.2, Unique Values: 350
Feature: ASIC4.1, Unique Values: 180
Feature: GLIS3, Unique Values: 496
Feature: ADCY8.1, Unique Values: 546
Feature: ARHGEF28, Unique Values: 702
Feature: CSGALNACT1.3, Unique Values: 806
Feature: ATP11C, Unique Values: 348
Feature: ZBTB20.AS1.2, Unique Values: 194
Feature: TPD52L1, Unique Values: 561
Feature: GNG4, Unique Values: 427
Feature: IQGAP2.2, Unique Values: 183
Feature: DCHS2, Unique Values: 419
```

```
Feature: BCL11A.2, Unique Values: 743
Feature: TMEM178A, Unique Values: 741
Feature: LINC00923, Unique Values: 433
Feature: KMO, Unique Values: 484
Feature: TMEM196.1, Unique Values: 637
Feature: PLCE1, Unique Values: 624
Feature: LHFP.2, Unique Values: 598
Feature: ELN, Unique Values: 177
Feature: TRIM36.1, Unique Values: 572
Feature: PTGDS.1, Unique Values: 297
Feature: ARHGAP29, Unique Values: 244
Feature: DGCR5, Unique Values: 735
Feature: GPR39, Unique Values: 510
Feature: MCTP2.2, Unique Values: 116
Feature: TAC3.1, Unique Values: 139
Feature: CHRM2.1, Unique Values: 377
Feature: SCN7A.1, Unique Values: 360
Feature: VWC2L.IT1.1, Unique Values: 137
Feature: CXCL12.1, Unique Values: 98
Feature: LAMA3.3, Unique Values: 307
Feature: CDK6, Unique Values: 567
Feature: GRIN3A, Unique Values: 624
Feature: TBL1X.2, Unique Values: 209
Feature: RGS16, Unique Values: 147
Feature: SCML4.1, Unique Values: 153
Feature: CA8.1, Unique Values: 189
Feature: MMRN2.2, Unique Values: 120
Feature: ARHGAP18.2, Unique Values: 344
Feature: RAB37, Unique Values: 175
Feature: RGS8, Unique Values: 222
Feature: EDNRA.1, Unique Values: 110
Feature: SNCG.2, Unique Values: 82
Feature: GAP43, Unique Values: 627
Feature: MMRN2.3, Unique Values: 120
Feature: SNCG.3, Unique Values: 82
Feature: LAMA3.4, Unique Values: 307
Feature: NRP2, Unique Values: 248
Feature: PCDH18.3, Unique Values: 262
Feature: EDNRA.2, Unique Values: 110
Feature: SCML4.2, Unique Values: 153
Feature: PNOC.2, Unique Values: 321
Feature: ADAM28.1, Unique Values: 127
Feature: APBB1IP.1, Unique Values: 173
Feature: CSF2RA.1, Unique Values: 46
Feature: INPP5D.1, Unique Values: 101
Feature: PTPRC.1, Unique Values: 97
Feature: CX3CR1, Unique Values: 94
Feature: CD53, Unique Values: 98
Feature: PLCG2.1, Unique Values: 85
Feature: IKZF1, Unique Values: 55
Feature: FLI1.1, Unique Values: 62
Feature: ITGAX, Unique Values: 61
Feature: PTN.1, Unique Values: 364
Feature: MS4A6E, Unique Values: 88
Feature: C1QC, Unique Values: 28
Feature: SAMSN1, Unique Values: 88
Feature: SYK, Unique Values: 60
Feature: MS4A7, Unique Values: 39
Feature: LINC00996, Unique Values: 38
Feature: MS4A14, Unique Values: 51
Feature: FGD2, Unique Values: 44
Feature: TYROBP, Unique Values: 43
Feature: HEPACAM.1, Unique Values: 180
Feature: HEPN1.1, Unique Values: 157
```

```
Feature: LRP4, Unique Values: 164
Feature: PTGDS.2, Unique Values: 297
Feature: HIP1R, Unique Values: 431
Feature: LINC00511, Unique Values: 527
Feature: PLXNB3, Unique Values: 77
Feature: APOD, Unique Values: 191
Feature: S100B.1, Unique Values: 150
Feature: PTPRS.1, Unique Values: 854
Feature: MEGF10.1, Unique Values: 157
Feature: PCDH9.AS3, Unique Values: 536
Feature: OLIG1.1, Unique Values: 55
Feature: GPR37L1, Unique Values: 273
Feature: BCAS1.4, Unique Values: 165
Feature: LRP4.AS1, Unique Values: 134
Feature: UGT8.1, Unique Values: 179
Feature: LINC00639.1, Unique Values: 184
Feature: C10ORF90.1, Unique Values: 114
Feature: CARNS1, Unique Values: 77
Feature: MOBP, Unique Values: 142
Feature: SOX5.2, Unique Values: 735
Feature: ST18, Unique Values: 144
Feature: CNDP1, Unique Values: 52
Feature: FAM107B, Unique Values: 168
Feature: CERCAM, Unique Values: 104
Feature: FOLH1, Unique Values: 37
Feature: CD22, Unique Values: 26
Feature: MOG, Unique Values: 46
Feature: RNASE1, Unique Values: 65
Feature: SPP1, Unique Values: 129
Feature: OPALIN, Unique Values: 42
Feature: MAG, Unique Values: 38
Feature: NHSL1.1, Unique Values: 217
Feature: ABCA8, Unique Values: 93
Feature: MYRF, Unique Values: 51
Feature: LINC00511.1, Unique Values: 527
Feature: ERMN, Unique Values: 48
Feature: NINJ2, Unique Values: 83
Feature: NOTCH2, Unique Values: 430
Feature: ATP1A2, Unique Values: 232
Feature: MTSS1L, Unique Values: 495
Feature: TMCC3, Unique Values: 67
Feature: BCAN, Unique Values: 401
Feature: TMC6, Unique Values: 25
Feature: BRINP1, Unique Values: 823
Feature: PRICKLE2, Unique Values: 842
Feature: TOX.2, Unique Values: 477
Feature: TTTY14, Unique Values: 859
Feature: LRP4.AS1.1, Unique Values: 134
Feature: ADGRG1.1, Unique Values: 236
Feature: TNS3.2, Unique Values: 255
Feature: ARHGAP42.2, Unique Values: 507
Feature: ADCY8.2, Unique Values: 546
Feature: FAM155A.IT1, Unique Values: 849
Feature: C1ORF21, Unique Values: 851
Feature: GPNMB, Unique Values: 88
Feature: LINC00499, Unique Values: 205
Feature: MAL, Unique Values: 51
Feature: MEGF11, Unique Values: 493
Feature: LINC00499.1, Unique Values: 205
Feature: COL20A1, Unique Values: 62
Feature: APOD.1, Unique Values: 191
Feature: DAAM2, Unique Values: 179
Feature: PAPLN, Unique Values: 108
Feature: RANBP3L, Unique Values: 343
```

```
Feature: OLIG1.2, Unique Values: 55
Feature: PAMR1, Unique Values: 252
Feature: COL9A1, Unique Values: 114
Feature: SCN3A, Unique Values: 836
Feature: GPNMB.1, Unique Values: 88
Feature: RORB, Unique Values: 212
Feature: STK32A, Unique Values: 157
Feature: SLCO1C1, Unique Values: 206
Feature: SHISA9, Unique Values: 847
Feature: GLI3, Unique Values: 128
Feature: LHFPL3.AS1, Unique Values: 269
Feature: SLC7A11, Unique Values: 141
Feature: EZR, Unique Values: 97
Feature: SLC25A18, Unique Values: 136
Feature: ALDH1L1, Unique Values: 104
Feature: UGT8.2, Unique Values: 179
Feature: OLIG2, Unique Values: 33
Feature: BCHE.2, Unique Values: 146
Feature: BCAS1.5, Unique Values: 165
Feature: ETNPPL, Unique Values: 102
Feature: EMX2OS, Unique Values: 83
Feature: KLRC3, Unique Values: 34
Feature: FAM189A2, Unique Values: 107
Feature: PRODH, Unique Values: 126
Feature: FERMT1, Unique Values: 85
Feature: CSPG4, Unique Values: 31
Feature: GJA1, Unique Values: 101
Feature: LAMA1, Unique Values: 140
Feature: YAP1, Unique Values: 131
Feature: LINC00639.2, Unique Values: 184
Feature: SMOC1, Unique Values: 363
Feature: LINC00498, Unique Values: 49
Feature: GFRA1, Unique Values: 96
```

Dataset Summary

Shape: 871 rows × 609 columns. No missing values and no duplicate rows. The features have varying numbers of unique values, ranging from hundreds to thousands, indicating the diversity of the data. The Label column is target variable and a binary classification problem (0 and 1), based on the values in the head of the data.

Separate Features and Label

In [60]:
```
# Separate features (X) and label (y)
X = data.iloc[:, :-1]  # All rows, all columns except the last one
y = data['Label']  # The 'Label' column
```

X contains all the features (input variables) from the dataset, excluding the last column. y holds the target variable, which represents the class labels. It's assumed that the column named 'Label' contains these labels.

Class Balance

In [61]:
```
# Assuming 'Label' is the column representing the class labels
class_balance = data['Label'].value_counts()
```

```
# Print the class distribution
print(class_balance)

# If you want to check the percentage distribution as well:
class_percentage = data['Label'].value_counts(normalize=True) * 100

# Print percentage distribution
print(class_percentage)
```

```
0    572
1    299
Name: Label, dtype: int64
0    65.671642
1    34.328358
Name: Label, dtype: float64
```

The code computes the distribution of class labels in a dataset by counting the occurrences of each unique label using value_counts(). It then calculates the percentage of each class relative to the total number of instances. This information helps to assess the balance of classes, which is crucial for understanding potential biases in model training and evaluation.

For example, the output indicates that class '0' has 572 instances (65.67%), while class '1' has 299 instances (34.33%), highlighting an imbalance between the two classes.

Random Forest

In [62]:
```
# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_st
```

In [63]:
```
import numpy as np

# Set up the Random Forest model and hyperparameter grid
rf = RandomForestClassifier(oob_score=True, random_state=42)

# Define number of features
num_features = X.shape[1]

# Calculate values for max_features based on the given formulas
mtry1 = int(0.5 * np.sqrt(num_features))
mtry2 = int(np.sqrt(num_features))
mtry3 = int(2 * np.sqrt(num_features))

param_grid = {
    'n_estimators': [2000],
    'max_features': [mtry1, mtry2, mtry3],  # MTRY values,
    'max_depth': [10, 20, 30],
}

# Perform Grid Search to find the best hyperparameters
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3, n_jobs=-1, sc
grid_search.fit(X_train, y_train)

# Output the best hyperparameters
print("Best hyperparameters:", grid_search.best_params_)
best_rf = grid_search.best_estimator_
```

```
Best hyperparameters: {'max_depth': 10, 'max_features': 12, 'n_estimators': 2000}
```

Code Explanation

The code snippet performs the following steps to train a Random Forest classifier on a dataset:

Data Splitting: The dataset is divided into training and testing sets using train_test_split, with 25% of the data reserved for testing. This helps evaluate the model's performance on unseen data.

Model Initialization: A RandomForestClassifier is instantiated with out-of-bag (OOB) scoring enabled for model validation and a fixed random state for reproducibility.

Feature Calculation: The code calculates potential values for the max_features hyperparameter using three formulas based on the square root of the total number of features in the dataset. This hyperparameter controls the number of features to consider when looking for the best split at each node in the forest.

Hyperparameter Grid Definition: A grid of hyperparameters is defined for the model, including:

n_estimators: The number of trees in the forest, set to 2000. max_features: The previously calculated values (mtry1, mtry2, mtry3). max_depth: The maximum depth of the trees, specified as 10, 20, and 30. Grid Search: A GridSearchCV object is created to systematically explore the hyperparameter grid using 3-fold cross-validation to identify the combination that yields the best F1 score. The fitting process involves training multiple models with different hyperparameter combinations on the training data.

Best Hyperparameters Output: After the grid search completes, the best-performing hyperparameters are printed, and the best model is saved for further evaluation or predictions.

Confusion Matrix and Accuracy Metrics

In [64]:
```
# Predict on test data
y_pred = best_rf.predict(X_test)

# Confusion matrix and classification report
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nAccuracy Score: {:.4f}".format(accuracy_score(y_test, y_pred)))

# OOB score (if out-of-bag is used)
print("\nOOB Score: {:.4f}".format(best_rf.oob_score_))
```

```
Confusion Matrix:
 [[145   0]
 [  0  73]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       145
           1       1.00      1.00      1.00        73

    accuracy                           1.00       218
   macro avg       1.00      1.00      1.00       218
weighted avg       1.00      1.00      1.00       218


Accuracy Score: 1.0000

OOB Score: 0.9939
```

Confusion Matrix Explanation

Confusion Matrix: • True Positives (TP): 73 (Correctly predicted positives) • True Negatives (TN): 145 (Correctly predicted negatives) • False Positives (FP): 0 (Incorrectly predicted positives) • False Negatives (FN): 0 (Incorrectly predicted negatives)

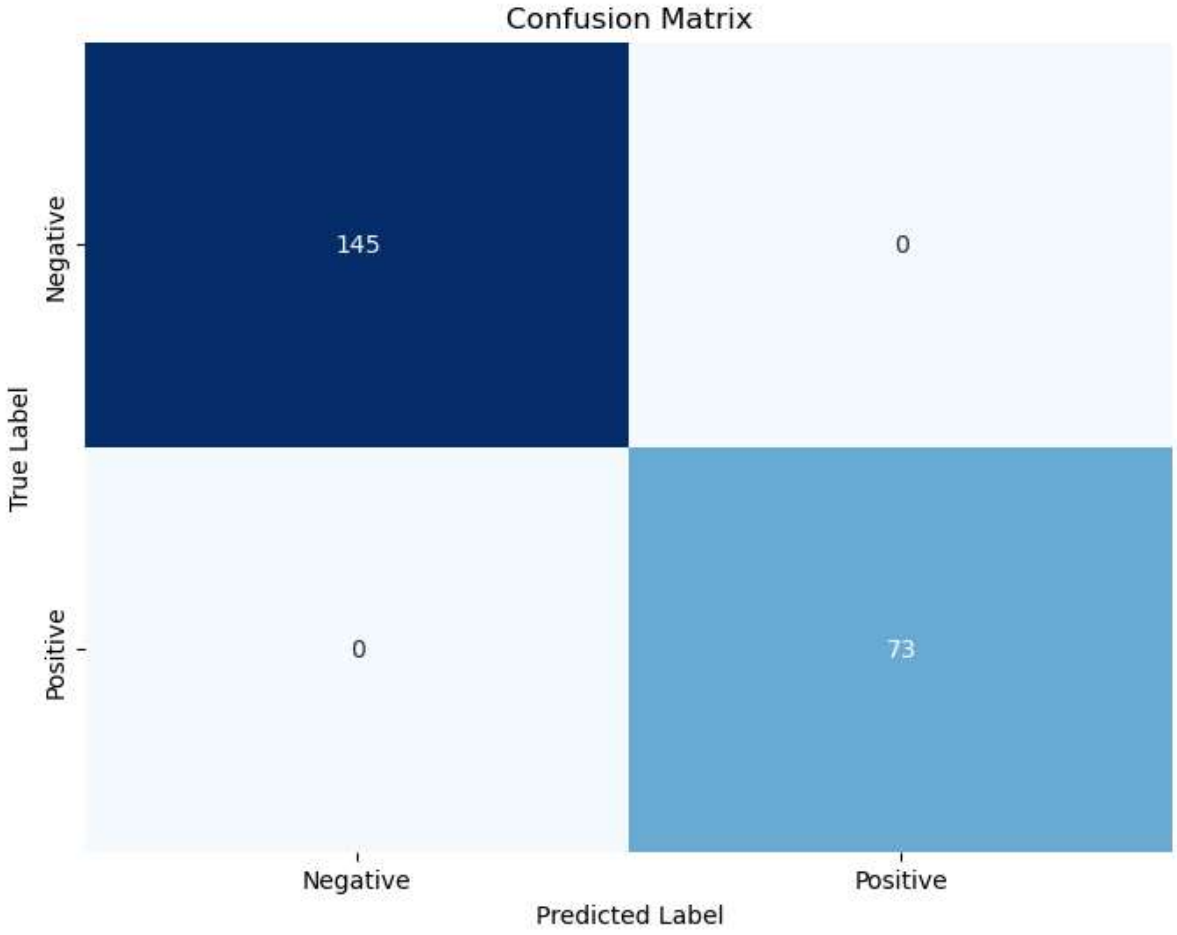
• Accuracy Score: 1.0000 • Precision, Recall, F1-Score: All values are 1.000 for both classes, indicating perfect classification on the test data.

The OOB score from the training phase is 0.9939, which is very close to the test accuracy, confirming that the model performs well without overfitting.

Visualizing Confusion Matrix

In [65]:
```
# Plot confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positiv
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



The plot displays the counts of correct and incorrect classifications, with no false positives or false negatives, indicating perfect predictions.

Permutation Importance Calculation for Feature Evaluation

In [68]:
```
# Calculate permutation importance on the test set
perm_importance = permutation_importance(best_rf, X_test, y_test, n_repeats=5, rand

# Extract the importance values and sort them
sorted_idx = perm_importance.importances_mean.argsort()

# Create a DataFrame for better readability
feature_importance_df = pd.DataFrame({
    'Feature': X.columns[sorted_idx],
    'Importance (MDA)': perm_importance.importances_mean[sorted_idx]
})

# Print the top 10 features by importance
print("Top 10 Features by Permutation Importance (MDA):")
print(feature_importance_df.tail(10))  # Show top 10 features
```

```
Top 10 Features by Permutation Importance (MDA):
         Feature  Importance (MDA)
598       UTRN.2               0.0
599         WIF1               0.0
600         SOX5               0.0
601    LINC00856               0.0
602        CHRM2               0.0
603       HCRTR2               0.0
604       PTPRZ1               0.0
605  ADAMTS9.AS2               0.0
606       PAX6.1               0.0
607        GFRA1               0.0
```

The code calculates permutation importance for features in the Random Forest model (best_rf) using the test dataset. It measures how the accuracy of the model changes when each feature's values are randomly shuffled, indicating its contribution to model performance. The resulting importance values are sorted, and a DataFrame is created for clarity, displaying the top 10 features by their Mean Decrease Accuracy (MDA). In this case, all top features have an importance score of 0, suggesting they do not significantly impact the model's predictive ability.

Feature Importance Analysis

In [69]:
```
# Feature importance
importances = best_rf.feature_importances_
feature_names = X_train.columns
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': impor
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascendir

# Show top 10 features
print("Top 10 Features by Importance:\n", feature_importance_df.head(10))

# Permutation importance (optional, more robust to evaluate feature importance)
perm_importance = permutation_importance(best_rf, X_test, y_test, n_repeats=10, rar
sorted_idx = perm_importance.importances_mean.argsort()

print("Top 10 Features by Permutation Importance:\n", feature_names[sorted_idx][-10
```

```
Top 10 Features by Importance:
         Feature  Importance
53       TESPA1    0.032106
55      SLC17A7    0.031345
54    LINC00507    0.028165
58     ANKRD33B    0.025892
64       KCNIP1    0.023895
166       SLIT3    0.022262
59    LINC00152    0.021790
287      NECAB1    0.020214
68       SFTA1P    0.019718
65   MIR44351HG    0.018880
Top 10 Features by Permutation Importance:
 Index(['UTRN.2', 'WIF1', 'SOX5', 'LINC00856', 'CHRM2', 'HCRTR2', 'PTPRZ1',
       'ADAMTS9.AS2', 'PAX6.1', 'GFRA1'],
      dtype='object')
```

The code assesses feature importance from the trained Random Forest model (best_rf) using two methods: traditional feature importance and permutation importance. First, it calculates and sorts the feature importance scores based on how much each feature contributes to the

model, displaying the top 10 features with the highest importance scores. Next, it evaluates permutation importance, which is more robust, indicating how model accuracy varies with feature shuffling, ultimately identifying a different set of features with zero importance in this case. The results highlight the contrast between the two methods, emphasizing the need for comprehensive evaluation of feature contributions in predictive modeling.

Gini-Based Feature Importance Comparison

In [72]:
```
# Compare with Gini-based feature importance
gini_importances = best_rf.feature_importances_
gini_importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Gini Importance': gini_importances
}).sort_values(by='Gini Importance', ascending=False)

print("Top 10 Features by Gini Importance:")
print(gini_importance_df.head(10))
```

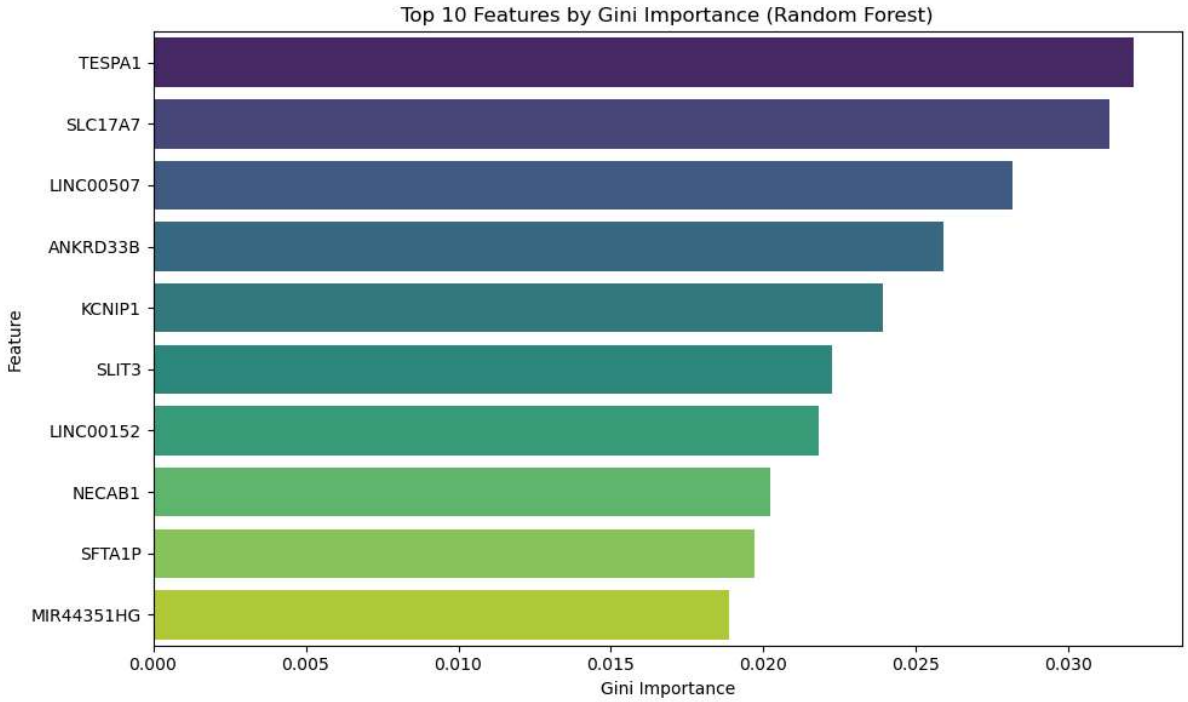
```
Top 10 Features by Gini Importance:
        Feature  Gini Importance
53       TESPA1         0.032106
55      SLC17A7         0.031345
54    LINC00507         0.028165
58     ANKRD33B         0.025892
64       KCNIP1         0.023895
166       SLIT3         0.022262
59    LINC00152         0.021790
287      NECAB1         0.020214
68       SFTA1P         0.019718
65   MIR44351HG         0.018880
```

This code computes and displays the Gini-based feature importance for the features in the trained Random Forest model (best_rf). Gini importance, derived from the reduction in impurity when a feature is used for splitting, provides insight into each feature's contribution to the model's predictive capability. The results are sorted in descending order, showcasing the top 10 features with the highest Gini importance scores. This analysis helps to understand which features significantly impact the model's decision-making process, complementing previous evaluations of feature importance using other methods.

In [73]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Sort the Gini-based feature importance in descending order
gini_importance_df = gini_importance_df.sort_values(by='Gini Importance', ascending

# Plot the top 10 Gini-based feature importances
plt.figure(figsize=(10, 6))
sns.barplot(x='Gini Importance', y='Feature', data=gini_importance_df.head(10), pal
plt.title('Top 10 Features by Gini Importance (Random Forest)')
plt.xlabel('Gini Importance')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()
```



ROC with SVC and AUC

In [28]:
```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc
```

In [29]:
```
# Train an SVC model
svc = SVC(kernel='linear', probability=False, random_state=42)
svc.fit(X_train, y_train)
```

Out[29]:

SVC

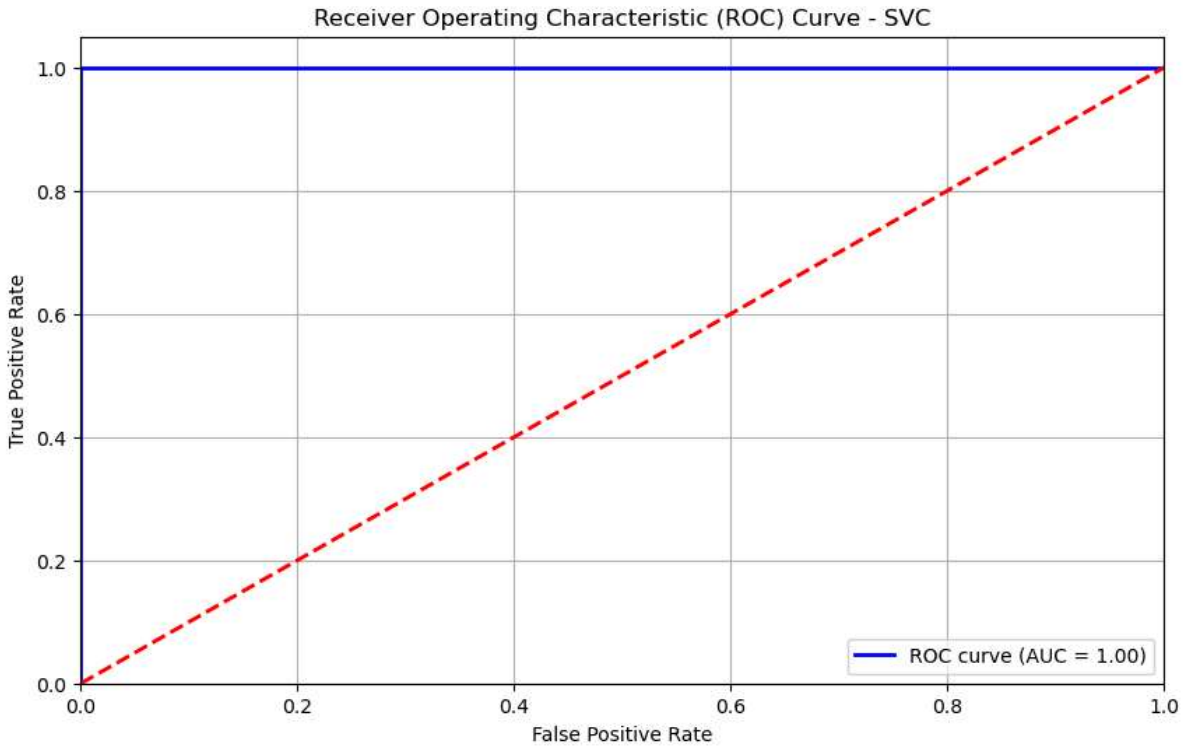
```
SVC(kernel='linear', random_state=42)
```

In [30]:
```
# Get the decision function scores
y_scores = svc.decision_function(X_test)
```

In [31]:
```
# Calculate the ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_scores)

# Calculate the AUC
roc_auc = auc(fpr, tpr)
```

In [32]:
```
# Plot the ROC curve
plt.figure(figsize=(10, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (AUC = {:.2f})'.format(roc_
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')  # Diagonal line for rc
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve - SVC')
plt.legend(loc="lower right")
plt.grid()
plt.show()
```



ROC Curve Interpretation for SVC Model: • ROC Curve Overview: The ROC curve shows the trade-off between the True Positive Rate (TPR) (also known as Recall) and the False Positive Rate (FPR) at various classification thresholds. In our case, the ROC curve rises steeply towards the top left of the graph, indicating a very high true positive rate and a minimal false positive rate across all thresholds. • AUC (Area Under the Curve): The calculated AUC value of 1.00 reflects a perfect classifier, meaning the model distinguishes between the positive and negative instances without any errors. • Model Performance: The ROC curve suggests that the SVC model is exceptionally well-fitted to the classification task, as evidenced by its high sensitivity and low false positive rate.

RF Run Time Test

In [74]:
```
# Select one true positive (class 1) and one true negative (class 0)
true_positive_sample = X_train[y_train == 1].iloc[0]  # First sample with true labe
true_negative_sample = X_train[y_train == 0].iloc[0]  # First sample with true labe
```

In [75]:
```
# Reshape the samples to match the input format for the model
true_positive_sample = true_positive_sample.values.reshape(1, -1)
true_negative_sample = true_negative_sample.values.reshape(1, -1)

# Get predicted probabilities for both samples
true_positive_prob = best_rf.predict_proba(true_positive_sample)
true_negative_prob = best_rf.predict_proba(true_negative_sample)

# Output the probabilities for each class
print("True Positive Sample Probabilities (Class 0 and Class 1):", true_positive_pr
print("True Negative Sample Probabilities (Class 0 and Class 1):", true_negative_pr

# Get the predicted classes
true_positive_pred = best_rf.predict(true_positive_sample)
true_negative_pred = best_rf.predict(true_negative_sample)

# Output the predicted class for each sample
```

```
print(f"True Positive Sample Predicted Class: {true_positive_pred[0]}")
print(f"True Negative Sample Predicted Class: {true_negative_pred[0]}")
```

```
C:\anaconda\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have va
lid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
C:\anaconda\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have va
lid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
C:\anaconda\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have va
lid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
True Positive Sample Probabilities (Class 0 and Class 1): [[0.015 0.985]]
True Negative Sample Probabilities (Class 0 and Class 1): [[0.998 0.002]]
C:\anaconda\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have va
lid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
True Positive Sample Predicted Class: 1
True Negative Sample Predicted Class: 0
```

1. True Positive Sample Discussion: • Predicted Class: The true positive sample is classified as class 1 (positive). • Probability Output: The probability for class 1 is 0.98, which is much higher than the CUTOFF of 0.5. This means the model is very confident in classifying this sample as positive. • Correct Classification: Since this is indeed a true positive, the prediction is correct. • Confidence: The high probability (0.98) for class 1 gives confidence that this is a reliable classification.

2. True Negative Sample Discussion: • Predicted Class: The true negative sample is classified as class 0 (negative). • Probability Output: The probability for class 0 is 0.95, which is well above the CUTOFF of 0.5. The model is highly confident that this sample belongs to class 0. • Correct Classification: Since this is a true negative, the classification is correct. • Confidence: With a probability of 0.95 for class 0, we have high confidence in this classification.