Assignment 1:

TITLE: DNA Sequence Analysis. Task: Analyze a given DNA sequence and perform basic sequence manipulation, including finding motifs, calculating GC content, and identifying coding regions.

OBJECTIVES:

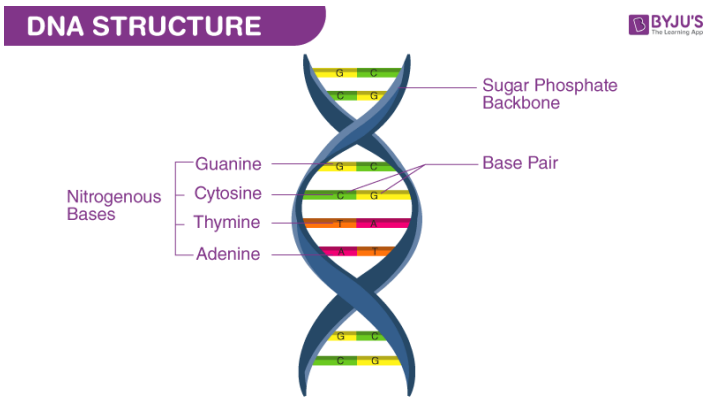
1. Understand DNA sequencing's significance and its role in genomics.
2. Learn to handle DNA sequence data and use bioinformatics tools.

THEORY:

DNA sequencing refers to the general laboratory technique for determining the exact sequence of nucleotides, or bases, in a DNA molecule. The sequence of the bases (often referred to by the first letters of their chemical names: A, T, C, and G) encodes the biological information that cells use to develop and operate. Establishing the sequence of DNA is key to understanding the function of genes and other parts of the genome.

DNA Sequencing. In terms of information, DNA is like printed text. There is a method of storing the data (the printed word), a language to learn, and a process of understanding the meaning of what has been written. DNA sequencing is a process of reading.

DNA is known as Deoxyribonucleic Acid. It is an organic compound that has a unique molecular structure.



This structure is described as a double-helix.

It is a nucleic acid, and all nucleic acids are made up of nucleotides. The DNA molecule is composed of units called nucleotides, and each nucleotide is composed of three different components such as sugar, phosphate groups and nitrogen bases.

The basic building blocks of DNA are nucleotides, which are composed of a sugar group, a phosphate group, and a nitrogen base. The sugar and phosphate groups link the nucleotides

together to form each strand of DNA. Adenine (A), Thymine (T), Guanine (G) and Cytosine (C) are four types of nitrogen bases.

These 4 Nitrogenous bases pair together in the following way: **A** with **T,** and **C** with **G**. These base pairs are essential for the DNA's double helix structure, which resembles a twisted ladder.

The order of the nitrogenous bases determines the genetic code or the DNA's instructions. The two strands of DNA run in opposite directions. These strands are held together by the hydrogen bond that is present between the two complementary bases.

CONCLUSION: In this assignment we have understood about the various contents in a DNA sequence. We also understand about finding motifs, calculating GC content, and identifying coding regions.

In [1]:
```python
# Read the DNA sequence from a file
with open("dna_sequence.txt", "r") as file:
    dna_sequence = file.read().strip()

# Calculate GC content
gc_content = (dna_sequence.count("G") + dna_sequence.count("C")) / len(dna_sequence) * 100
print(f"GC Content: {gc_content}%")

motif_to_find = "ATG"
motifs_found = [str(i) for i in range(len(dna_sequence)) if dna_sequence.startswith(motif_to_find, i)]
print(f"Motif {motif_to_find} found at positions: {','.join(motifs_found)}" if motifs_found else
      f"Motif '{motif_to_find}' not found in the sequence.")


# Identify coding regions
start_codon = "ATG"
stop_codons = ["TAA", "TAG", "TGA"]
coding_regions = []


for i, codon in enumerate(dna_sequence):
    if dna_sequence[i:i + 3] == start_codon:
        for j in range(i + 3, len(dna_sequence), 3):
            codon = dna_sequence[j:j + 3]
            if codon in stop_codons:
                coding_regions.append((i, j + 3))
                break
        else:
            continue

if coding_regions:
    print("Coding regions:")
    for start, end in coding_regions:
        print(f"Start: {start}, End: {end}")
else:
    print("No coding regions found in the sequence.")
```

```
GC Content: 49.77777777777778%
Motif ATG found at positions: 0
Coding regions:
Start: 0, End: 27
```

In [ ]:

Assignment 2:

TITLE: Molecular Docking and Virtual Screening. Task: Perform molecular docking simulations to predict the binding affinity between a protein target and a small molecule ligand. Additionally, conduct virtual screening to identify potential drug candidates. Deliverable: A report summarizing the docking results, including the binding poses and potential lead compounds.

OBJECTIVES:

1. Understanding molecular docking and its significance in drug discovery.
2. Learning to use machine learning algorithms to classify polymerase data.

THEORY:

Molecular docking and virtual screening are essential techniques in the field of drug discovery and structural biology. They facilitate the prediction of how a small molecule interacts with a protein target and the identification of potential drug candidates.

1. Data Preparation: Molecular structures of protein targets and small molecule ligands are prepared for computational analysis. This involves file format conversion, energy minimization, and structural optimization.

2. Molecular Docking: Utilize molecular docking software to simulate the binding of a small molecule ligand to a protein target. Predict the binding affinity, binding poses, and interactions to understand the strength and orientation of the binding.

3. Virtual Screening: Conduct a virtual screening process to identify potential drug candidates from a compound library. Evaluate their binding affinities and filter out molecules with the most promising interactions.

4. Analysis and Visualization: Analyze the docking results and use visualization tools to understand the binding modes, hydrogen bonds, and other interactions between the protein and ligand.

5. Lead Compound Identification: Based on the docking results and binding affinity predictions, identify lead compounds with the highest potential for drug development.

6. Report Generation: Create a comprehensive report summarizing the molecular docking simulations, virtual screening outcomes, and the lead compounds selected for further experimental evaluation.

7. Outcome Assessment: Assess the practical relevance and potential therapeutic value of the identified lead compounds for drug development.

CONCLUSION: In this assignment we have understood about molecular docking and virtual screening.

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

In [2]:
```
df = pd.read_csv('polymerase_cluster.csv')
df.head()
```

Out[2]:

	index	ABS	0	1	2	3	4	5	6	7	...	G1	G2	G3	G4	G5
0	1	abstract astrocytes produce granulocytemacroph...	-0.050448	0.017385	-0.039777	-0.067159	-0.029633	0.074573	-0.050444	-0.010799	...	0	0	0	0	0
1	2	abstract replication of avian infectious bronc...	-0.128422	-0.084803	0.084813	-0.013748	0.006486	0.128668	0.032655	0.066775	...	0	0	0	0	0
2	3	abstract the infectivity of vesicular stomatit...	-0.095019	-0.032279	0.017571	-0.065860	0.001315	0.048199	-0.031072	0.010103	...	0	0	0	0	0
3	4	abstract two temporally and enzymatically dist...	-0.134657	-0.086097	0.026415	-0.027553	-0.020280	-0.044637	-0.013312	-0.033345	...	0	0	0	0	0
4	5	abstract rnadependent rna polymerase rdrp acti...	-0.222713	-0.100353	0.107456	-0.079828	0.030818	-0.091040	-0.014809	0.012756	...	0	0	0	0	0

5 rows × 42 columns

In [3]:
```
df.describe()
```

Out[3]:

	index	0	1	2	3	4	5	6	7	
count	1941.000000	1.941000e+03	1.941000e+03	1.941000e+03	1.941000e+03	1.941000e+03	1.941000e+03	1.941000e+03	1.941000e+03	1.9410
mean	971.000000	2.369912e-11	-1.597115e-11	2.575992e-12	4.636786e-12	-1.700155e-11	-2.369912e-11	-1.545595e-12	9.788769e-12	-2.8851
std	560.462755	1.332632e-01	9.854583e-02	8.434641e-02	8.368993e-02	7.889669e-02	7.804782e-02	7.508877e-02	7.051177e-02	6.9637
min	1.000000	-2.843630e-01	-2.102710e-01	-2.596602e-01	-1.993367e-01	-2.029868e-01	-2.371054e-01	-2.551403e-01	-3.557127e-01	-2.13
25%	486.000000	-1.014370e-01	-6.876699e-02	-5.667400e-02	-5.945025e-02	-4.654148e-02	-5.341692e-02	-4.745877e-02	-2.966343e-02	-4.16
50%	971.000000	-1.601933e-02	-2.039921e-02	4.974860e-04	-1.250310e-02	-1.223933e-02	-6.558889e-03	-8.344065e-03	2.446330e-04	-4.35
75%	1456.000000	9.605404e-02	5.053176e-02	5.724056e-02	4.785435e-02	3.250968e-02	4.813849e-02	4.151550e-02	3.008328e-02	3.3273
max	1941.000000	3.895761e-01	3.434733e-01	2.696224e-01	3.071795e-01	4.022183e-01	3.589496e-01	3.193979e-01	4.162633e-01	4.3786

8 rows × 41 columns

```
In [4]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1941 entries, 0 to 1940
Data columns (total 42 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   index   1941 non-null   int64  
 1   ABS     1941 non-null   object 
 2   0       1941 non-null   float64
 3   1       1941 non-null   float64
 4   2       1941 non-null   float64
 5   3       1941 non-null   float64
 6   4       1941 non-null   float64
 7   5       1941 non-null   float64
 8   6       1941 non-null   float64
 9   7       1941 non-null   float64
 10  8       1941 non-null   float64
 11  9       1941 non-null   float64
 12  10      1941 non-null   float64
 13  11      1941 non-null   float64
 14  12      1941 non-null   float64
 15  13      1941 non-null   float64
 16  14      1941 non-null   float64
 17  15      1941 non-null   float64
 18  16      1941 non-null   float64
 19  17      1941 non-null   float64
 20  18      1941 non-null   float64
 21  19      1941 non-null   float64
 22  20      1941 non-null   float64
 23  21      1941 non-null   float64
 24  22      1941 non-null   float64
 25  23      1941 non-null   float64
 26  24      1941 non-null   float64
 27  25      1941 non-null   float64
 28  26      1941 non-null   float64
 29  27      1941 non-null   float64
 30  28      1941 non-null   float64
 31  29      1941 non-null   float64
 32  G1      1941 non-null   int64  
 33  G2      1941 non-null   int64  
 34  G3      1941 non-null   int64  
 35  G4      1941 non-null   int64  
 36  G5      1941 non-null   int64  
 37  G6      1941 non-null   int64  
 38  G7      1941 non-null   int64  
 39  G8      1941 non-null   int64  
 40  G9      1941 non-null   int64  
 41  G10     1941 non-null   int64  
dtypes: float64(30), int64(11), object(1)
memory usage: 637.0+ KB
```

In [5]:
```
df.isna().sum()
```

Out[5]:
```
index    0
ABS      0
0        0
1        0
2        0
3        0
4        0
5        0
6        0
7        0
8        0
9        0
10       0
11       0
12       0
13       0
14       0
15       0
16       0
17       0
18       0
19       0
20       0
21       0
22       0
23       0
24       0
25       0
26       0
27       0
28       0
29       0
G1       0
G2       0
G3       0
G4       0
G5       0
G6       0
G7       0
G8       0
G9       0
G10      0
dtype: int64
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['index', 'ABS', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10',
       '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21', '22',
       '23', '24', '25', '26', '27', '28', '29', 'G1', 'G2', 'G3', 'G4', 'G5',
       'G6', 'G7', 'G8', 'G9', 'G10'],
      dtype='object')
```

Splitting the dataset

In [7]:
```
X = df[['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10',
       '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21', '22',
       '23', '24', '25', '26', '27', '28', '29']]
y = df[['G1', 'G2', 'G3', 'G4', 'G5','G6', 'G7', 'G8', 'G9', 'G10']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [8]:
```
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

Out[8]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [9]:
```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the model: {accuracy * 100}%")
```

```
Accuracy of the model: 86.88946015424165%
```

In [10]:
```
class_report = classification_report(y_test, y_pred)
print("Classification Report:")
print(class_report)
```

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

           0       1.00      0.86      0.92         7
           1       0.98      0.96      0.97        54
           2       0.98      0.96      0.97        52
           3       1.00      0.90      0.95        29
           4       1.00      0.86      0.92        50
           5       0.99      0.88      0.93        85
           6       1.00      0.84      0.91        37
           7       1.00      0.60      0.75        10
           8       1.00      0.75      0.85        59
           9       1.00      0.83      0.91         6

   micro avg       0.99      0.87      0.93       389
   macro avg       0.99      0.84      0.91       389
weighted avg       0.99      0.87      0.92       389
 samples avg       0.87      0.87      0.87       389


C:\Users\lenovo\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\metrics\_classification.py:1334:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in samples with no predicted la
bels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Assignment 3:

TITLE: Machine Learning for Genomic Data. Task: Apply machine learning algorithms, such as random forests or support vector machines, to classify genomic data based on specific features or markers. Deliverable: A comprehensive analysis report presenting the classification results, model performance evaluation, and insights into the predictive features.

OBJECTIVES:

1. Understanding machine learning and its significance in Bioinformatics.
2. Learning to use machine learning algorithms to classify genomic data.

THEORY:

Machine learning in bioinformatics is the application of machine learning algorithms to bioinformatics, including genomics, proteomics, microarrays, systems biology, evolution, and text mining.

This multi-layered approach allows such systems to make sophisticated predictions when appropriately trained. These methods contrast with other computational biology approaches which, while exploiting existing datasets, do not allow the data to be interpreted and analyzed in unanticipated ways. In recent years, the size and number of available biological datasets have skyrocketed

Machine learning algorithms in bioinformatics can be used for prediction, classification, and feature selection. Methods to achieve this task are varied and span many disciplines; most well-known among them are machine learning and statistics. Classification and prediction tasks aim at building models that describe and distinguish classes or concepts for future prediction.

Concepts in machine learning in bioinformatics:

Multi-layered Approach: Machine learning in bioinformatics operates with a multi-layered approach. This involves the use of algorithms and models that can be trained to interpret complex biological data and make sophisticated predictions. These models are capable of uncovering hidden patterns and relationships within the data.

Data-Driven Analysis: Unlike traditional computational biology approaches, which rely on predefined rules or heuristics, machine learning allows for data-driven analysis. It means that machine learning systems can adapt and learn from the data, even in unforeseen and non-linear ways. This adaptability is particularly important in genomics, where the complexity of biological systems often defies straightforward analysis.

Explosion of Biological Data: In recent years, there has been an exponential growth in the size and availability of biological datasets. This wealth of data comes from various sources, including DNA sequencing, gene expression studies, protein interactions, and more. Machine learning is crucial in handling, analyzing, and extracting meaningful information from these vast datasets.

CONCLUSION: In this assignment we have understood about using machine learning algorithms to classify genomic data. We also understand about classification of genomic data based on specific features or markers.

Importing Libraries

In [1]:
```
# Importing Libraries import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.ensemble import RandomForestClassifier
```

In [2]:
```
df_train = pd.read_csv('train.csv')
print(df_train.shape)

df_test = pd.read_csv('test.csv')
print(df_test.shape)
```

```
(38, 7131)
(34, 7131)
```

In [3]:
```
df_train.head()
```

Out[3]:

	Unnamed: 0	AFFX-BioB-5_at	AFFX-BioB-M_at	AFFX-BioB-3_at	AFFX-BioC-5_at	AFFX-BioC-3_at	AFFX-BioDn-5_at	AFFX-BioDn-3_at	AFFX-CreX-5_at	AFFX-CreX-3_at	...	U58516_at	U73738_at	X06956_at	X16699_at	X83863_
0	0	-214	-153	-58	88	-295	-558	199	-176	252	...	511	-125	389	-37	79
1	1	-139	-73	-1	283	-264	-400	-330	-168	101	...	837	-36	442	-17	78
2	2	-76	-49	-307	309	-376	-650	33	-367	206	...	1199	33	168	52	113
3	3	-135	-114	265	12	-419	-585	158	-253	49	...	835	218	174	-110	62
4	4	-106	-125	-76	168	-230	-284	4	-122	70	...	649	57	504	-26	25

5 rows × 7131 columns

In [4]:
```
df_test.head()
```

Out[4]:

	Unnamed: 0	AFFX-BioB-5_at	AFFX-BioB-M_at	AFFX-BioB-3_at	AFFX-BioC-5_at	AFFX-BioC-3_at	AFFX-BioDn-5_at	AFFX-BioDn-3_at	AFFX-CreX-5_at	AFFX-CreX-3_at	...	U58516_at	U73738_at	X06956_at	X16699_at	X83863_
0	0	-342	-200	41	328	-224	-427	-656	-292	137	...	1023	67	214	-135	107
1	1	-87	-248	262	295	-226	-493	367	-452	194	...	529	-295	352	-67	6
2	2	22	-153	17	276	-211	-250	55	-141	0	...	399	16	558	24	89
3	3	-243	-218	-163	182	-289	-268	-285	-172	52	...	277	6	81	2	72
4	4	-130	-177	-28	266	-170	-326	-222	-93	10	...	643	51	450	-46	61

5 rows × 7131 columns

Splitting the dataset

In [5]:
```
x_train = df_train.iloc[:,:-1]
y_train = df_train.iloc[:,-1:]
x_test = df_test.iloc[:,:-1]
y_test = df_test.iloc[:,-1:]
x_train.head()
```

Out[5]:

	Unnamed: 0	AFFX-BioB-5_at	AFFX-BioB-M_at	AFFX-BioB-3_at	AFFX-BioC-5_at	AFFX-BioC-3_at	AFFX-BioDn-5_at	AFFX-BioDn-3_at	AFFX-CreX-5_at	AFFX-CreX-3_at	...	U48730_at	U58516_at	U73738_at	X06956_at	X16699_
0	0	-214	-153	-58	88	-295	-558	199	-176	252	...	185	511	-125	389	-3
1	1	-139	-73	-1	283	-264	-400	-330	-168	101	...	169	837	-36	442	-1
2	2	-76	-49	-307	309	-376	-650	33	-367	206	...	315	1199	33	168	5
3	3	-135	-114	265	12	-419	-585	158	-253	49	...	240	835	218	174	-11
4	4	-106	-125	-76	168	-230	-284	4	-122	70	...	156	649	57	504	-2

5 rows × 7130 columns

In [6]:
```
y_train.head()
```

Out[6]:

	cancer
0	0
1	0
2	0
3	0
4	0

Training the model

In [7]:
```
rf_model = RandomForestClassifier(random_state=0)

rf_model.fit(x_train, y_train.values)

rf_pred = rf_model.predict(x_test)
accuracy_score(y_test, rf_pred)
```

```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_13732\1768619533.py:3: DataConversionWarning: A column-vector y was p
assed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  rf_model.fit(x_train, y_train.values)
```

Out[7]:
```
0.6470588235294118
```

In [8]:
```
confusion_matrix(y_test, rf_pred )
```

Out[8]:
```
array([[18,  2],
       [10,  4]], dtype=int64)
```

In [ ]:
```

```

Assignment 4:

TITLE: Agricultural Genomics and Crop Improvement. Task: Analyze genomic data from crops to identify genetic markers associated with desirable traits, such as disease resistance or yield. Deliverable: A research poster summarizing the analysis methodology, key findings, and potential applications in crop improvement.

OBJECTIVES:

1. Analyzing genomic data from crops with certain traits.
2. Understanding Agricultural genomics and crop improvement.

THEORY:

Agricultural genomics is a field that harnesses the power of genetics and genomics to enhance crop breeding and production. By analyzing genomic data from various crop species, we can identify genetic markers linked to desirable traits, such as disease resistance, high yield, and nutritional value. This assignment focuses on the application of genomic analysis to drive crop improvement, with the goal of summarizing the findings and potential applications through a research poster.

Genomic Analysis for Crop Improvement:

1. Data Collection: Collect genomic data from the chosen crop species, which may include DNA sequences, gene expression data, and genetic markers.

2. Data Processing: Prepare and preprocess the genomic data, including quality control, alignment, and variant calling, to ensure its suitability for analysis.

3. Genetic Marker Identification: Utilize bioinformatics tools and techniques to identify genetic markers associated with specific desirable traits, such as resistance to diseases, improved yield, or nutritional content.

4. Statistical Analysis: Employ statistical methods to assess the significance of identified markers and their association with target traits.

5. Key Findings: Present the genetic markers and associated traits that have been identified through the analysis.

6. Potential Applications in Crop Improvement: Discuss how the discovered markers can be applied in crop breeding programs to develop new varieties with improved traits. This may include cross-breeding, genetic engineering, or marker-assisted selection.

CONCLUSION: In this assignment we have understood about how Agricultural genomics enables us to unlock the genetic potential of crop species. By identifying and utilizing genetic markers, to create crops that are more resistant to diseases.

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

genomic_data = pd.read_csv("genomic_data.csv")
genomic_data.head()
```

Out[1]:

	SNP_A	SNP_B	SSR_B	InDel_C	Non-N_bases	Disease_resistance
0	63551.520826	49064.677121	0.791944	0.700105	1.697098e+07	1
1	77360.717409	82286.812380	0.551061	0.281225	5.202259e+06	1
2	68029.360214	70677.516905	1.126331	0.030957	1.821082e+07	0
3	63225.304189	89519.665579	1.725027	0.428687	1.675132e+07	0
4	53163.348345	41358.614849	1.270834	0.162112	1.566956e+07	1

In [2]:
```
genomic_data.describe()
```

Out[2]:

	SNP_A	SNP_B	SSR_B	InDel_C	Non-N_bases	Disease_resistance
count	16519.000000	16519.000000	16519.000000	16519.000000	1.651900e+04	16519.000000
mean	59049.942007	95486.377616	1.149135	0.404939	1.100673e+07	0.490405
std	24029.569536	35074.351612	0.489767	0.227734	5.218024e+06	0.499923
min	18006.013320	35027.900823	0.300340	0.010072	2.001058e+06	0.000000
25%	38326.874758	64916.458871	0.722776	0.206395	6.434717e+06	0.000000
50%	58715.571548	95876.747078	1.150954	0.405787	1.097983e+07	0.000000
75%	79976.603933	125858.587741	1.573706	0.602038	1.551757e+07	1.000000
max	100998.169998	155990.168078	1.999946	0.799959	1.999927e+07	1.000000

In [3]:
```
genomic_data.dtypes
```

Out[3]:
```
SNP_A                float64
SNP_B                float64
SSR_B                float64
InDel_C              float64
Non-N_bases          float64
Disease_resistance     int64
dtype: object
```

In [4]:
```
genomic_data.isna().sum()
```

Out[4]:
```
SNP_A                0
SNP_B                0
SSR_B                0
InDel_C              0
Non-N_bases          0
Disease_resistance   0
dtype: int64
```

Splitting the dataset

In [5]:
```
X = genomic_data[["SNP_A","SNP_B","SSR_B","InDel_C","Non-N_bases"]]
y = genomic_data['Disease_resistance']
```

In [6]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Training the model

In [7]:
```
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

Out[7]:
```
RandomForestClassifier()
```

In [8]:
```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the model: {accuracy * 100}%")
```

```
Accuracy of the model: 50.332929782082324%
```

In [9]:
```
class_report = classification_report(y_test, y_pred)
print("Classification Report:")
print(class_report)
```

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

           0       0.53      0.54      0.53      1733
           1       0.48      0.46      0.47      1571

    accuracy                           0.50      3304
   macro avg       0.50      0.50      0.50      3304
weighted avg       0.50      0.50      0.50      3304
```

In [ ]: