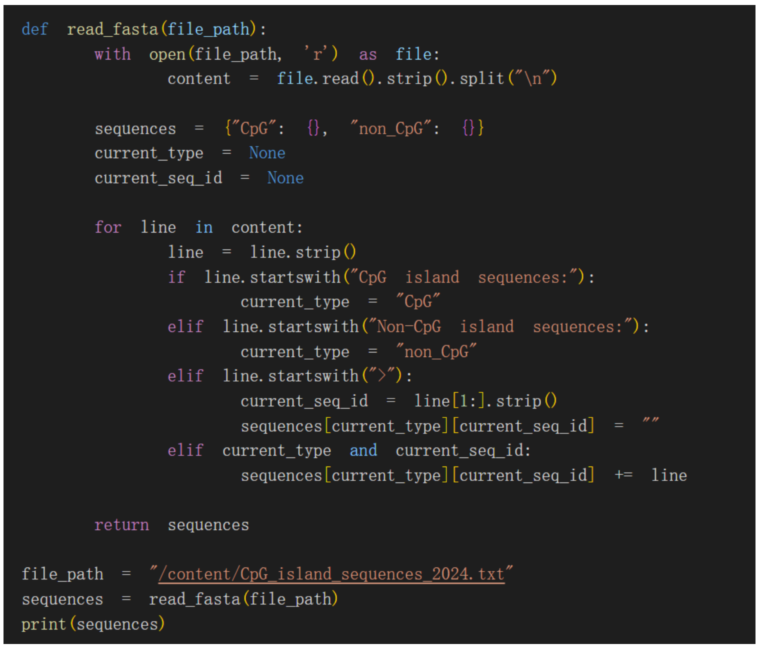
1. Introduction

The goal is to classify whether this sequence **("GCACCGGAGTCCGGAGAACTGAAGTCCGCCAAGACAAATTAGTTCACTGTA GAAGGGCTCCGAGGGA")** is CpG island or non-CpG island using log-odds ratios and Markov chains.

2. Methods

Firstly, I transformed the doc file into a text file and used Python to store the CpG and non-CpG island in two dictionaries.



```python
def read_fasta(file_path):
    with open(file_path, 'r') as file:
        content = file.read().strip().split("\n")

    sequences = {"CpG": {}, "non_CpG": {}}
    current_type = None
    current_seq_id = None

    for line in content:
        line = line.strip()
        if line.startswith("CpG island sequences:"):
            current_type = "CpG"
        elif line.startswith("Non-CpG island sequences:"):
            current_type = "non_CpG"
        elif line.startswith(">"):
            current_seq_id = line[1:].strip()
            sequences[current_type][current_seq_id] = ""
        elif current_type and current_seq_id:
            sequences[current_type][current_seq_id] += line

    return sequences

file_path = "/content/CpG_island_sequences_2024.txt"
sequences = read_fasta(file_path)
print(sequences)
```

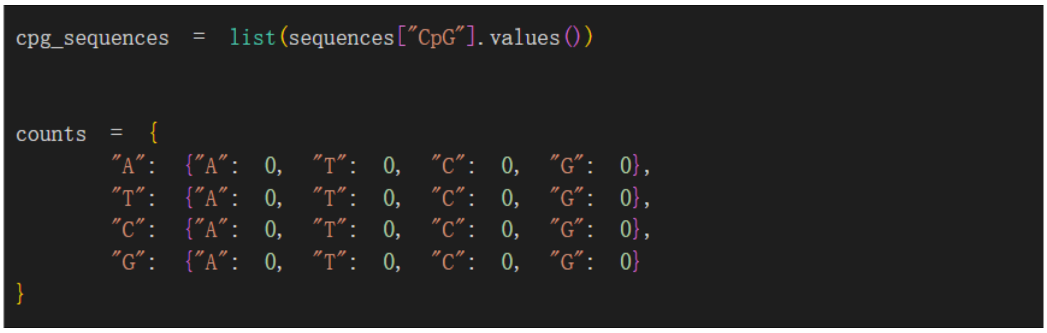
- **CpG island training sequences analysis**

<u>**Transition counts**</u>

A nested dictionary [counts] is initialized to store the count of each possible nucleotide transition:

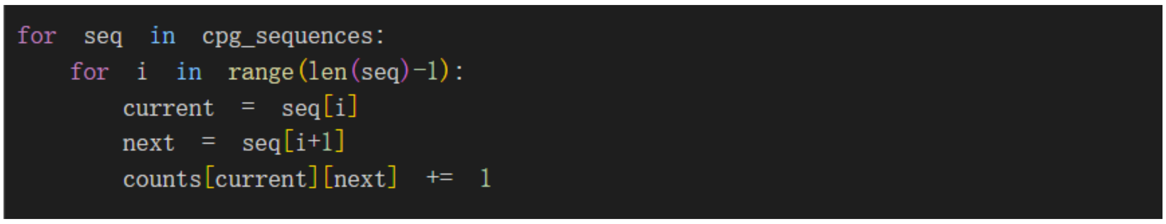
The outer key: current nucleotide

The inner key: next nucleotide



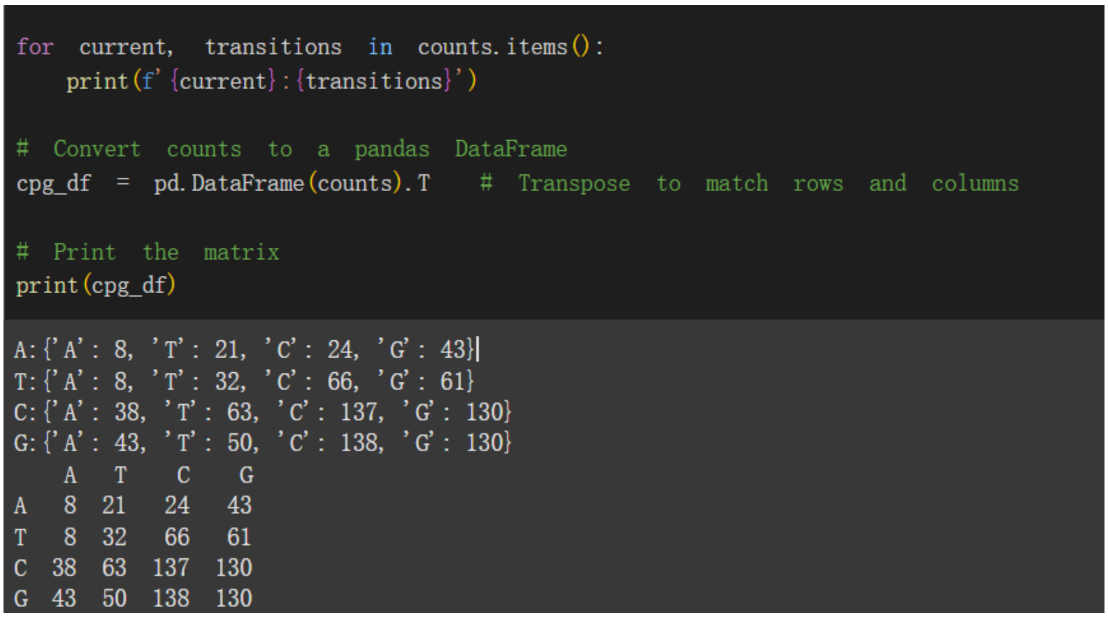
```python
cpg_sequences = list(sequences["CpG"].values())


counts = {
    "A": {"A": 0, "T": 0, "C": 0, "G": 0},
    "T": {"A": 0, "T": 0, "C": 0, "G": 0},
    "C": {"A": 0, "T": 0, "C": 0, "G": 0},
    "G": {"A": 0, "T": 0, "C": 0, "G": 0}
}
```

This part of the code iterates through each CpG sequence and then goes through each nucleotide pair in the sequence (For each pair of nucleotides: **current** → **next**). The [counts] dictionary is then updated to increment the count for that specific transition.



```
for seq in cpg_sequences:
    for i in range(len(seq)-1):
        current = seq[i]
        next = seq[i+1]
        counts[current][next] += 1
```

After processing all sequences, [counts] contains the total number of times each nucleotide transition occurs in the CpG sequences. I also converted the [counts] dictionary to a data frame by using pandas to gain a clear visualization.

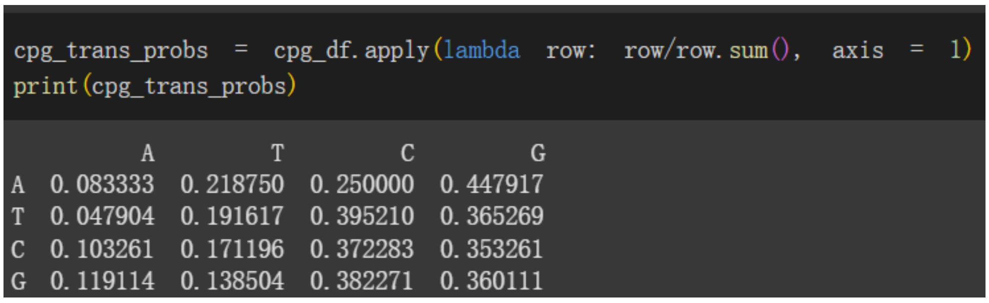


```
for current, transitions in counts.items():
    print(f'{current}:{transitions}')

# Convert counts to a pandas DataFrame
cpg_df = pd.DataFrame(counts).T  # Transpose to match rows and columns

# Print the matrix
print(cpg_df)
```

```
A:{'A': 8, 'T': 21, 'C': 24, 'G': 43}
T:{'A': 8, 'T': 32, 'C': 66, 'G': 61}
C:{'A': 38, 'T': 63, 'C': 137, 'G': 130}
G:{'A': 43, 'T': 50, 'C': 138, 'G': 130}
    A   T    C    G
A   8  21   24   43
T   8  32   66   61
C  38  63  137  130
G  43  50  138  130
```

Transition probabilities

This line of code computes the transition probabilities for each nucleotide transition in the CpG sequences using the transition counts stored in the `cpg_df` data frame, which was created in the previous step. I calculated the sum of each row and then divided each specific transition count by that sum to obtain the transition probability. As required by the Markov chain model, the transition probabilities in each row sum to 1.



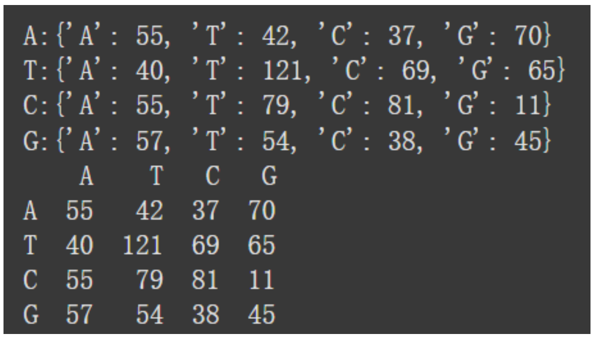
```
cpg_trans_probs = cpg_df.apply(lambda row: row/row.sum(), axis = 1)
print(cpg_trans_probs)
```

```
          A         T         C         G
A  0.083333  0.218750  0.250000  0.447917
T  0.047904  0.191617  0.395210  0.365269
C  0.103261  0.171196  0.372283  0.353261
G  0.119114  0.138504  0.382271  0.360111
```

- **Non CpG island training sequences analysis**

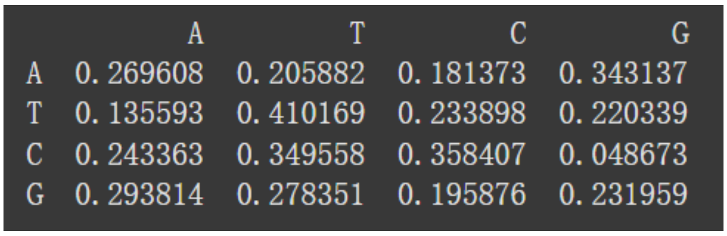
The way I processed the non-CpG island training sequences is the same as that of the CpG island sequences. I will simply show the results here.

Transition counts



```
A:{'A': 55, 'T': 42, 'C': 37, 'G': 70}
T:{'A': 40, 'T': 121, 'C': 69, 'G': 65}
C:{'A': 55, 'T': 79, 'C': 81, 'G': 11}
G:{'A': 57, 'T': 54, 'C': 38, 'G': 45}
    A    T   C   G
A  55   42  37  70
T  40  121  69  65
C  55   79  81  11
G  57   54  38  45
```

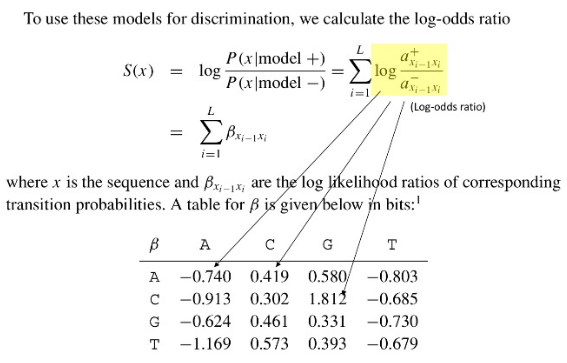
Transition probabilities



```
          A         T         C         G
A  0.269608  0.205882  0.181373  0.343137
T  0.135593  0.410169  0.233898  0.220339
C  0.243363  0.349558  0.358407  0.048673
G  0.293814  0.278351  0.195876  0.231959
```

The C→G transition probabilities are very different between CpG island sequences and non-CpG island sequences. (CpG island sequences: 0.35, non-CpG island sequences:0.05).

- **Log-odds ratio**



To use these models for discrimination, we calculate the log-odds ratio

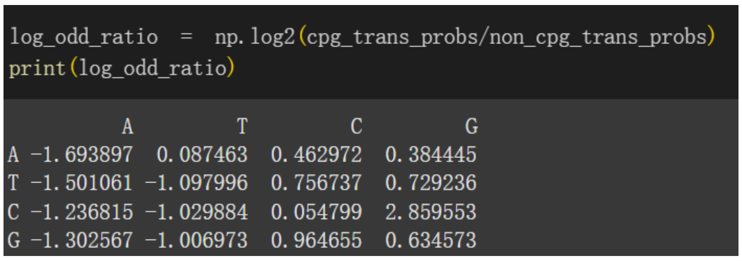
$$S(x) = \log \frac{P(x|\text{model}+)}{P(x|\text{model}-)} = \sum_{i=1}^{L} \log \frac{a^{+}_{x_{i-1}x_i}}{a^{-}_{x_{i-1}x_i}} = \sum_{i=1}^{L} \beta_{x_{i-1}x_i}$$

(Log-odds ratio)

where x is the sequence and $\beta_{x_{i-1}x_i}$ are the log likelihood ratios of corresponding transition probabilities. A table for β is given below in bits:[1]

β	A	C	G	T
A	−0.740	0.419	0.580	−0.803
C	−0.913	0.302	1.812	−0.685
G	−0.624	0.461	0.331	−0.730
T	−1.169	0.573	0.393	−0.679

Based on the material, the log odds ratio is transition probabilities in CpG island divided by transition probabilities in non-CpG island.



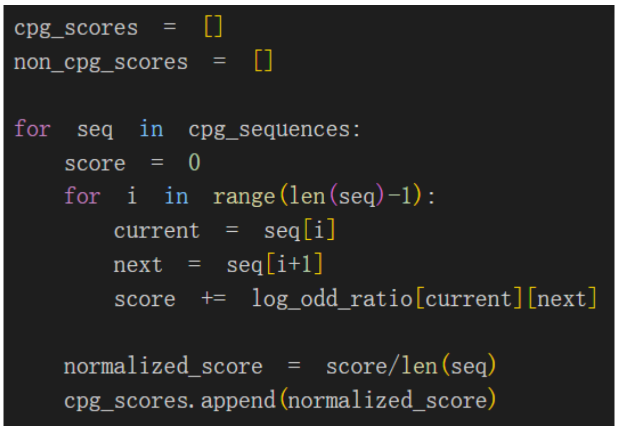
```
log_odd_ratio = np.log2(cpg_trans_probs/non_cpg_trans_probs)
print(log_odd_ratio)
```

```
          A         T         C         G
A -1.693897  0.087463  0.462972  0.384445
T -1.501061 -1.097996  0.756737  0.729236
C -1.236815 -1.029884  0.054799  2.859553
G -1.302567 -1.006973  0.964655  0.634573
```

- **Count the normalized scores**

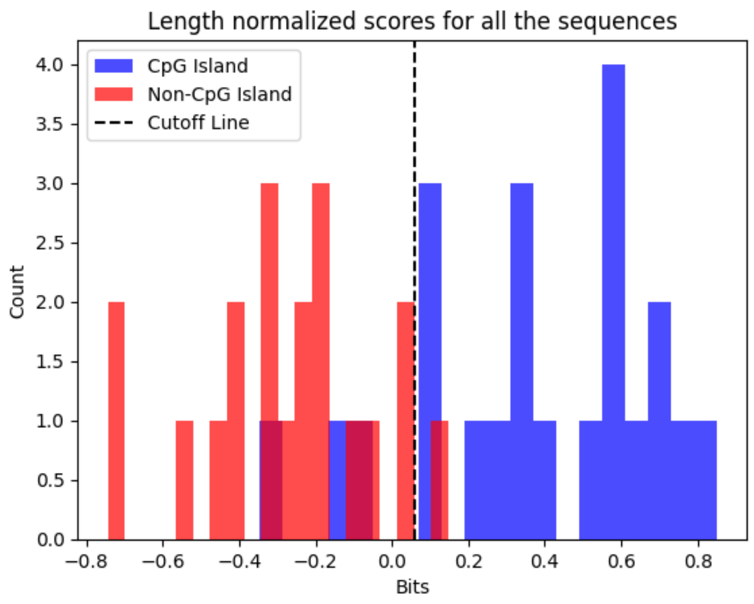
After analyzing the transition probabilities for both types of sequences and getting the log-odd ratio, we can then count the normalized score for each CpG and non-CpG island sequence.

I initialized two lists to store the scores of CpG and non-CpG island sequences individually. The outer loop iterates through each sequence in 'cpg_sequences' (CpG training sequences), while the inner loop iterates through each nucleotide pair in the sequence (current → next). For each transition, the log-odds ratio from the matrix is added to the cumulative score. After processing the entire sequence, the score is normalized by **dividing it by the sequence length**. For non-CpG sequences, the whole process is identical to the CpG loop but operates on the 'non_cpg_sequences' (non-CpG training sequences).



```
cpg_scores = []
non_cpg_scores = []

for seq in cpg_sequences:
    score = 0
    for i in range(len(seq)-1):
        current = seq[i]
        next = seq[i+1]
        score += log_odd_ratio[current][next]

    normalized_score = score/len(seq)
    cpg_scores.append(normalized_score)
```

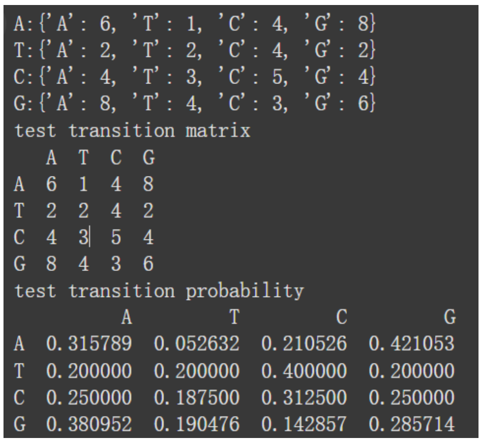
Next, I used Matplotlib to visualize the normalized scores of CpG island and non-CpG island sequences. I set the cutoff line at **x = 0.06** to minimize the overlap between CpG and non-CpG scores.



- **Prediction**

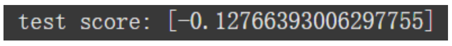
  For the test sequence, I used the same method to obtain the transition counts, transition probabilities, and the normalized score.

  Transition counts and probabilities



```
A:{'A': 6, 'T': 1, 'C': 4, 'G': 8}
T:{'A': 2, 'T': 2, 'C': 4, 'G': 2}
C:{'A': 4, 'T': 3, 'C': 5, 'G': 4}
G:{'A': 8, 'T': 4, 'C': 3, 'G': 6}
test transition matrix
   A  T  C  G
A  6  1  4  8
T  2  2  4  2
C  4  3  5  4
G  8  4  3  6
test transition probability
          A         T         C         G
A  0.315789  0.052632  0.210526  0.421053
T  0.200000  0.200000  0.400000  0.200000
C  0.250000  0.187500  0.312500  0.250000
G  0.380952  0.190476  0.142857  0.285714
```

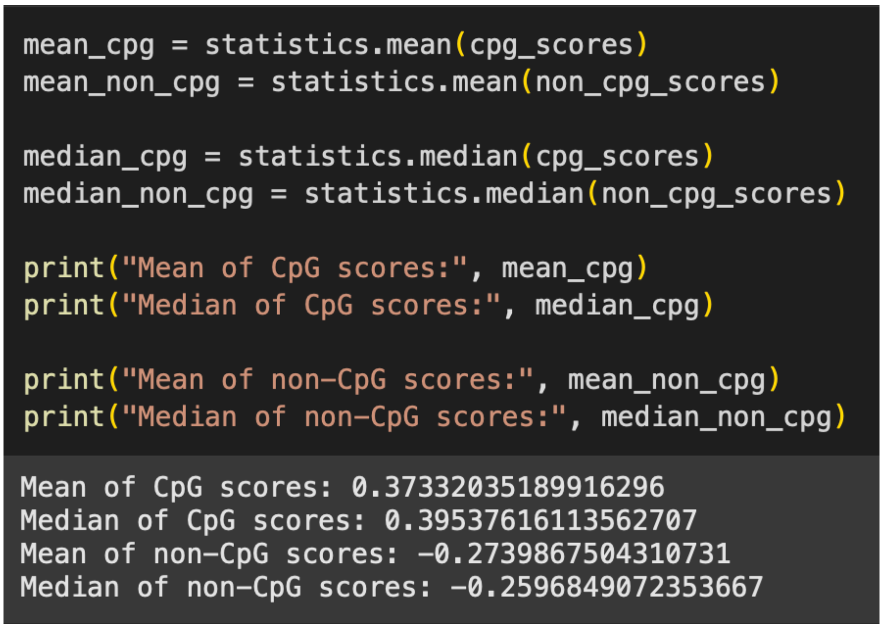
  Normalized score



```
test score: [-0.12766393006297755]
```

Statistics analysis

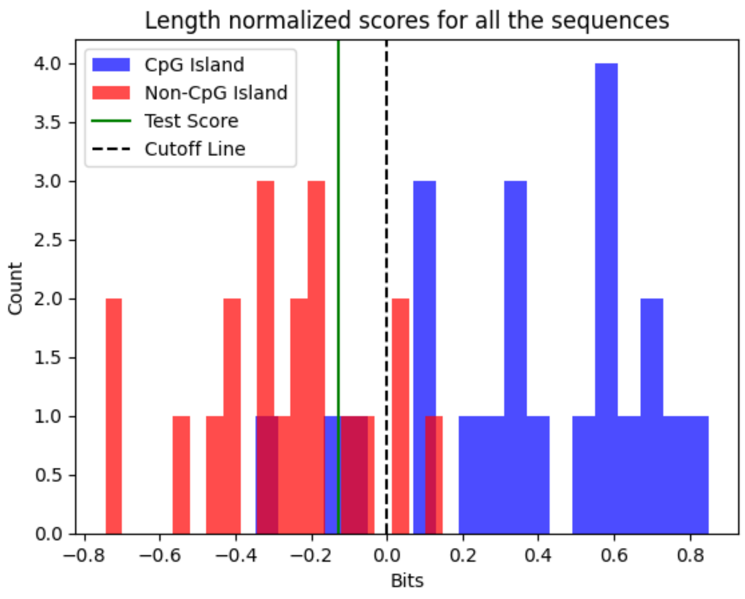
I also calculated the mean value and the median of both CpG scores and non CpG scores.



```
mean_cpg = statistics.mean(cpg_scores)
mean_non_cpg = statistics.mean(non_cpg_scores)

median_cpg = statistics.median(cpg_scores)
median_non_cpg = statistics.median(non_cpg_scores)

print("Mean of CpG scores:", mean_cpg)
print("Median of CpG scores:", median_cpg)

print("Mean of non-CpG scores:", mean_non_cpg)
print("Median of non-CpG scores:", median_non_cpg)
```

```
Mean of CpG scores: 0.37332035189916296
Median of CpG scores: 0.39537616113562707
Mean of non-CpG scores: -0.2739867504310731
Median of non-CpG scores: -0.2596849072353667
```

From the calculation, we can observe that the mean and median of CpG scores are roughly 0.4, while the non-CpG scores are around -0.27. Based on this result, I assume that the test sequence is more likely to be a non-CpG sequence.**Histogram**



Finally, I plotted the test score in the histogram, represented by the green line. We can see that the test score is to the left of the cutoff line. In conclusion, the test sequence is a non-CpG island based on my analysis.