Sequence Classification Model Report

Nabila Nabil
Caroline Emad El-Dein
Viola Fawzy
Maria Atef

27/5/2022

—

Artificial Intelligence

—

Prepared for

Professor / Abdel-Rahman Hedar

1. Introduction

12] A promoter is a DNA region where the transcription process of a gene starts. It controls the binding of RNA polymerase which transcribes DNA to mRNA that is eventually translated into a functioning protein.
In other words, the promoter region is responsible for controlling when and where in the organism a specific gene is expressed.

Studying and analyzing Promoters has been extremely important in understanding many different diseases especially the ones of genetic nature and the reason for that is:
 that a 4] several of these diseases are caused by mutations or variations in the promoter region _and not in the coding region as previously thought_ for example:

- Asthma 67]
- Diabetes [8]
- Beta thalassemia [9]
- Huntington's disease [10]
- Cancer [8]

[5] which will open a pathway to correct that specific variance leading to the treatment of the resulting disease by interference (editing) of the structure or the number of promoter-bound proteins leaving the other genes that are associative with the target gene intact

[5][4]That's why the classification of sequences to promoters and non-promoters in increasingly gaining the interest and attention of many bioinformatics researchers and due to their importance, there was several studies dedicated to classifying promoters and non- promoters regions, but the opportunity for further improvement is present.

To Achieve this We considered using:

- Deep Learning (Neural Networks) **:-**
  Issues with that approach were:
  >Inconvenient in Hardware as it's (Hardware-Dependent)
  >Not being able to show the problem to the network in an easy and accurate way
  >The time duration isn't known

- Machine learning:

1. support vector machine **:-**
   Issues with that approach were:
   >Time complexity was remarkably high and gave the approximately the same results as the perceptron

2. k-Nearest Neighbor classifier model **:-**
   Issues with that approach were:
   >Lower accuracy (59.4%)
   >Sensitive to noise and missing data (Nan)

3. Random Forest classifier model
   Issues with that approach were:
   >lower accuracy (68.33%)
   >inconvenient storage and memories usage

4. Tree classifier model
   Issues with that approach were:
   >lower accuracy (62%)

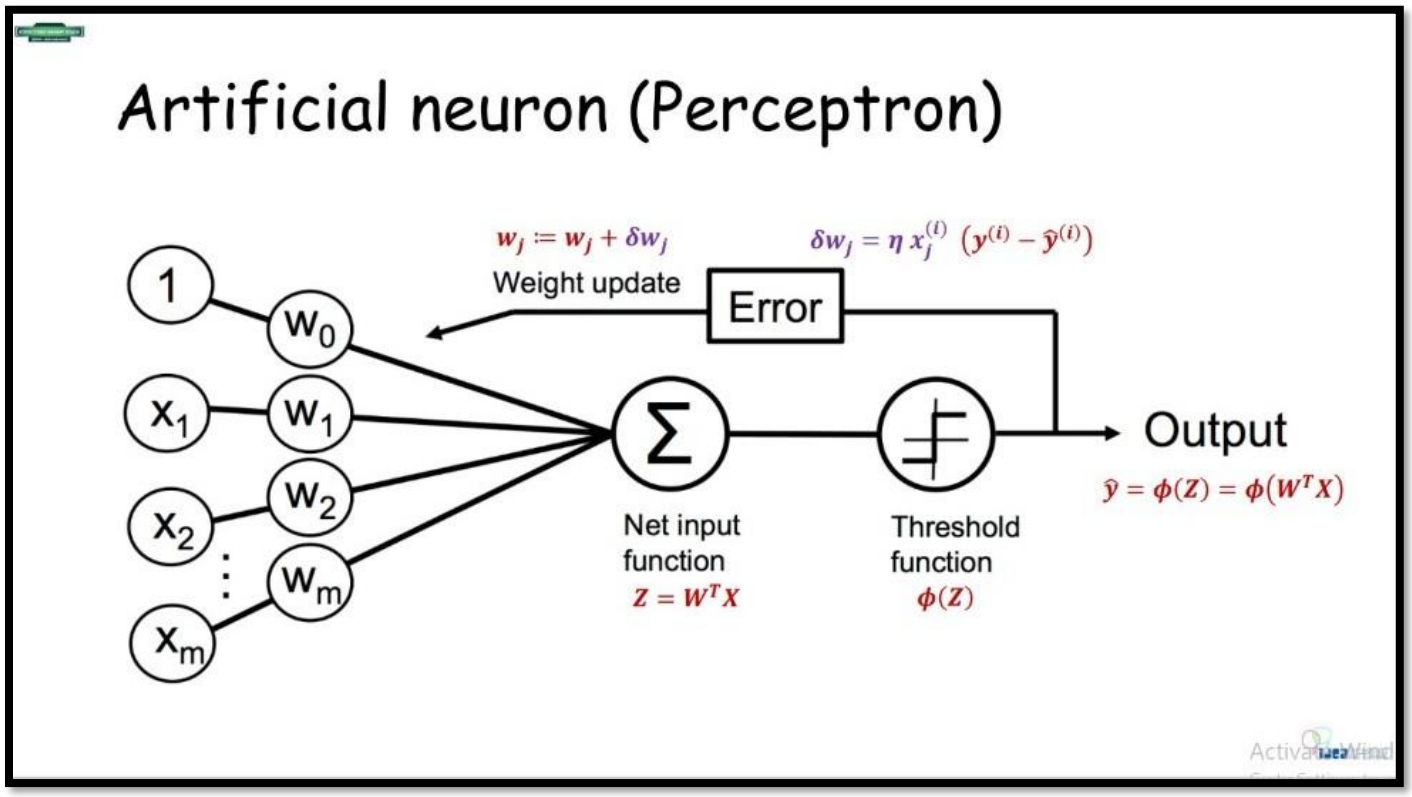
eventually we opted for:

- Perceptron classifier model

Which gave the best results in the fastest time

2. Methodology:

1. Perceptron Algorithm:
   [11][12]
   it is linear machine learning(a supervised learning) algorithm for making binary classification (promoter and non-promoter)



it lets artificial neurons learn and process certain data in the training set one at a time.

The Perceptron algorithm learns the weights for the input signals so it can draw a linear boundary

which let us to identify between the two linearly separable classes Non promoter (0) and promoter(1).

Perceptron Learning Rule:

-Perceptron Learning Rule states that the algorithm automatically learns the optimal weight coefficients.
-Then The input features are multiplied with these weights to determine wither a neuron fires or not.

-The Perceptron receives several inputs, and if the sum of the input signals surpasses a certain threshold, it either outputs a signal or does not output anything .

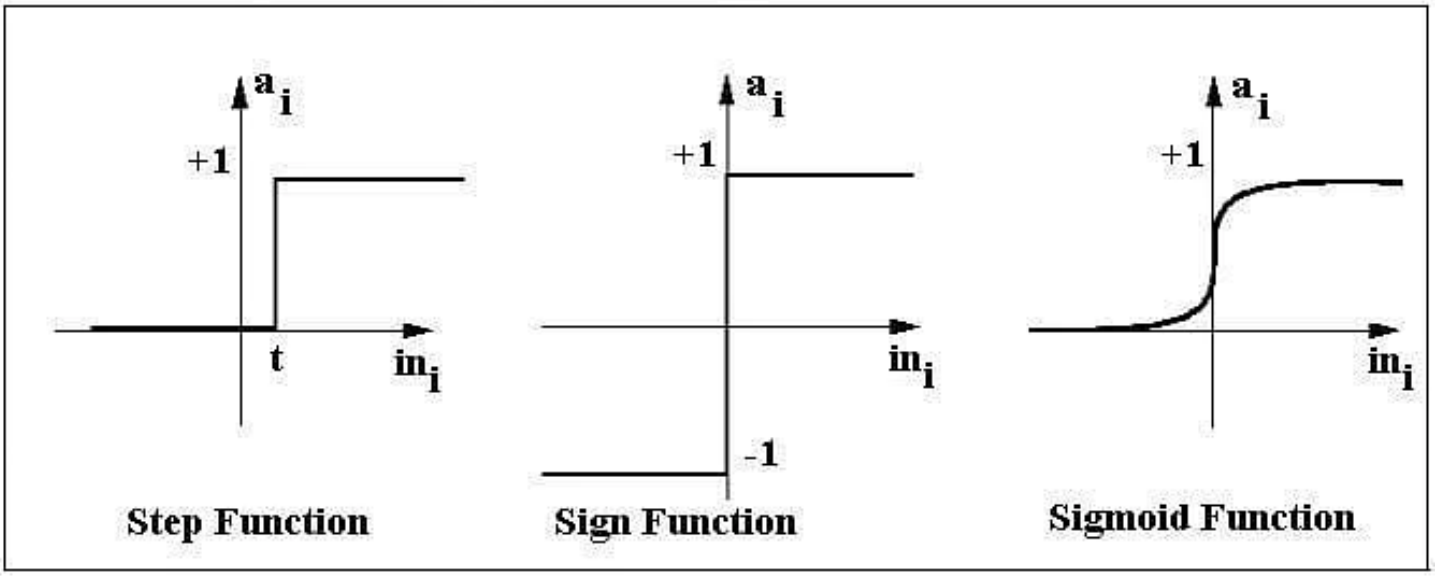
that's how the perceptron (after supervised learning and classification) can be used to predict the label of a sample

Perceptron main Function:

Perceptron is a function that maps its input “x,” after being multiplied with the learning weight coefficient; then generates an output value ”f(x)”

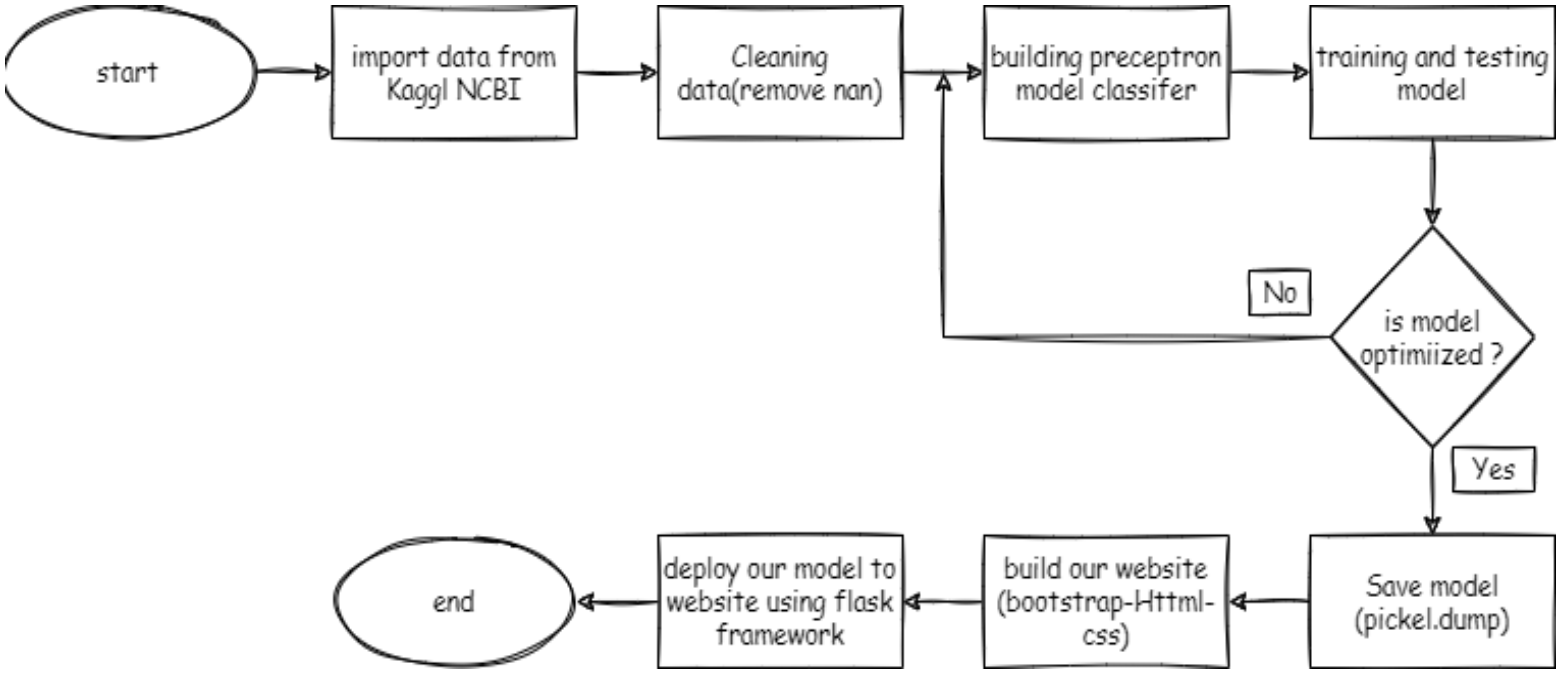
Activation Functions of Perceptron:

The activation function has a step rule (change the output into +1 or -1) to check whether or not the output of the weighting function is greater than zero



Step function is triggered above a specific value of the output; else it outputs 0 . Sign Function outputs +1 or -1 depending on if or not the neuron output is greater than 0 . Sigmoid function outputs a value between 0 and 1.

2.Project Flowchart:



3. Pseudocode:

[13]

```
BEGIN
Import pandas as pd and numpy libraries as np from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.linear_model import Perceptron
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer

Df1=pd.read(promoter dataset text)
Df1.dropna(subset= [‘unnamed:0’],how=all,inplace=true)
df1.reset_index(inplace = True)
df1.drop(['EP 1 (+) mt:Col_1; range -400 to -100.', 'index'], axis = 1, inplace=True)
df1.rename(columns={'Unnamed: 0': "sequence"}, inplace = True)
df1['label'] = 0

same thinge in promoter dataset
df = pd.concat([df1, df2], axis = 0 )
for seq in df['sequence']:
begin
   if 'N' in seq:
begin

      display(df.loc[df['sequence'] == seq])
end
end
```

```
df.drop([1822], inplace = True)
df3=df
display df3

begin
def Kmers_funct(seq, size=4):
   return [seq[x:x+size].lower() for x in range(len(seq) - size + 1)]
end

df['words'] = df.apply(lambda x: Kmers_funct(x['sequence']), axis=1)
df = df.drop('sequence', axis=1)
Y = df.iloc[:, 0].values
X = list(df['words'])
for item in range(len(X)):
begin
   X[item] = ' '.join(X[item])
end

from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(ngram_range=(4,4))
X = cv.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.20,random_state=1)

ppn = Perceptron(eta0=.01, max_iter=75)
ppn.fit(X_train, y_train)
df3['words'] = df3.apply(lambda x: Kmers_funct(x['sequence']), axis=1)
df3 = df3.drop('sequence', axis=1)
X2 = list(df3['words'])

for item in range(len(X2)):
begin
   X2[item] = ' '.join(X2[item])
end

cv = CountVectorizer(ngram_range=(4,4))
X2 = cv.fit_transform(X2)
Display ppn.predict(X2[0])
import pickle

pickle.dump(ppn,open('model.pkl','wb'))
model=pickle.load(open('model.pkl','rb'))

 END
```

3. Experimental Simulation

1. Python programming language:
as it is open-source language that includes a huge library collection that makes it and best language in dealing with machine learning convenience wise.

2.Perceptron learning algorithm:
as it archived the highest accuracy with the lowest run time as it is a linear classifier with time complexity O(t)

3.Biopython to work with sequences
4.Pandas and NumPy libraries for cleaning the dataset
5.Sklearn for:

- training, testing, and splitting the dataset
- calculating accuracy score
- import models

6.Kaggle and NCBI for acquiring our dataset
7.Google Collab editor for accessing faster GPUs, TPUs and more RAM.
8.Bootstrap (html, CSS, JavaScript) to create a web page
9.Flask framework to link our model to the web page

Test cases: We used we use 20% from dataset in test and 80% in train.

Code details:

1. import pandas and numpy library
2. Reading non-promoter data and cleaning it from missing data

```
[ ]  import pandas as pd
     import numpy as np
     import os

     df1 = pd.read_csv('/content/drive/MyDrive/ML/NonPromoterSequence.txt', sep = '>', )
     df1.dropna(subset=['Unnamed: 0'], how='all', inplace=True)
     df1.reset_index(inplace = True)
     df1.drop(['EP 1 (+) mt:CoI_1; range -400 to -100.', 'index'], axis = 1, inplace=True)
     df1.rename(columns={'Unnamed: 0': "sequence"}, inplace = True)
     df1['label'] = 0
```

3. Reading promoter data and cleaning it from missing data

```
df2 = pd.read_csv('/content/drive/MyDrive/ML/PromoterSequence.txt', sep = '>', )
df2.dropna(subset=['Unnamed: 0'], how='all', inplace=True)
df2.reset_index(inplace = True)
df2.drop(['EP 1 (+) mt:CoI_1; range -100 to 200.', 'index'], axis = 1, inplace=True)
df2.rename(columns={'Unnamed: 0': "sequence"}, inplace = True)
df2['label'] = 1
```

4. Link two dataset together in one dataframe.

```
[ ] df = pd.concat([df1, df2], axis = 0 )
    df.shape
```

```
(22600, 2)
```

5. Function to Divide Data to 4-mer
6. Calling the Kmers_funct and dividing the data

```
[ ] def Kmers_funct(seq, size=4):
        return [seq[x:x+size].lower() for x in range(len(seq) - size + 1)]
```

```
[ ]
    df['words'] = df.apply(lambda x: Kmers_funct(x['sequence']), axis=1)
    df = df.drop('sequence', axis=1)
```

```
df
df['words']
```

```
0        [taat, aatt, atta, ttac, taca, acat, catt, att...
1        [attt, tttt, tttt, ttta, ttac, taca, acaa, caa...
2        [agag, gaga, agat, gata, atag, tagg, aggt, ggt...
3        [tatg, atgt, tgta, gtat, tata, atat, tata, ata...
4        [agaa, gaaa, aaat, aata, ataa, taat, aata, ata...
                               ...
11295    [cgac, gaca, acaa, caaa, aaag, aagt, agtt, gtt...
11296    [cata, atat, tatc, atct, tcta, ctac, taca, aca...
11297    [atac, tacc, accg, ccgc, cgcg, gcgg, cgga, gga...
11298    [atta, ttat, tatt, attc, ttcc, tccg, ccga, cga...
11299    [aatt, attc, ttca, tcat, catt, attt, ttta, tta...
Name: words, Length: 22598, dtype: object
```

7. Converting Charcters to int64 matrcies using CountVectorizer

```
[ ] Y = df.iloc[:, 0].values
```

```
[ ] X = list(df['words'])
    for item in range(len(X)):
        X[item] = ' '.join(X[item])
```

```
[ ] from sklearn.feature_extraction.text import CountVectorizer
    cv = CountVectorizer(ngram_range=(4,4)) #The n-gram size of 4 is previously determined by testing
    X = cv.fit_transform(X)
```

8. Importing sklearn
9. Splitting the Data into Test and Train

```
import numpy as np
import pandas as pd

from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

from sklearn.linear_model import Perceptron
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble  import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

```
[ ] # Splitting the human dataset into the training set and test set
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X,
                                                        Y,
                                                        test_size = 0.20,
                                                        random_state=1)
```

10. Testing the model prediction on a single sequence
11. Importing Pickle and saving the model

```
[ ] df3.at[0,'sequence']="ATTTTTACAAGAACAAGACATTTAACTTTAACTTTATCTTTAGCTTTACCTTTATGATTATGTTTTATATTATATGGATG
```

```
[ ]
    df3['words'] = df3.apply(lambda x: Kmers_funct(x['sequence']), axis=1)
    df3 = df3.drop('sequence', axis=1)
```

```
[ ] X2 = list(df3['words'])
    for item in range(len(X2)):
        X2[item] = ' '.join(X2[item])
```

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(ngram_range=(4,4)) #The n-gram size of 4 is previously determined by testing
X2 = cv.fit_transform(X2)
```

+ Code + Text

```
[ ] ppn.predict(X2[0])

    array([0])
```

```
[ ] import pickle
    pickle.dump(ppn,open('model.pkl','wb'))
    model=pickle.load(open('model.pkl','rb'))
```

4.Results and Technical Discussion:

1. dataset after cleaning from nan

```
df
```

	sequence	label
0	TAATTACATTATTTTTTTATTTACGAATTTGTTATTCCGCTTTTAT...	0
1	ATTTTTACAAGAACAAGACATTTAACTTTAACTTTATCTTTAGCTT...	0
2	AGAGATAGGTGGGTCTGTAACACTCGAATCAAAAACAATATTAAGA...	0
3	TATGTATATAGAGATAGGCGTTGCCAATAACTTTTGCGTTTTTTGC...	0
4	AGAAATAATAGCTAGAGCAAAAAACAGCTTAGAACGGCTGATGCTC...	0
...	...	...
11295	CGACAAAGTTTGATCCATGTGCATTCTTGGCGCCTTATCGATAGCT...	1
11296	CATATCTACATCTCGCTTGCTCCTTCCCTTTCGCTGCGTGTGTGTG...	1
11297	ATACCGCGGAAGCGCAAAAGTACCAGAATTTCCCTGGTATCGCGCT...	1
11298	ATTATTCCGAATTCTTTTATCAGATTTAAATATGGGAAACACTTTA...	1
11299	AATTCATTTATACCTGCATTTGTAACTGTACTAAATCTTCAACCAA...	1

22598 rows × 2 columns

2. Training the Perceptron
3. Printing the accuracy score in both testing and training da

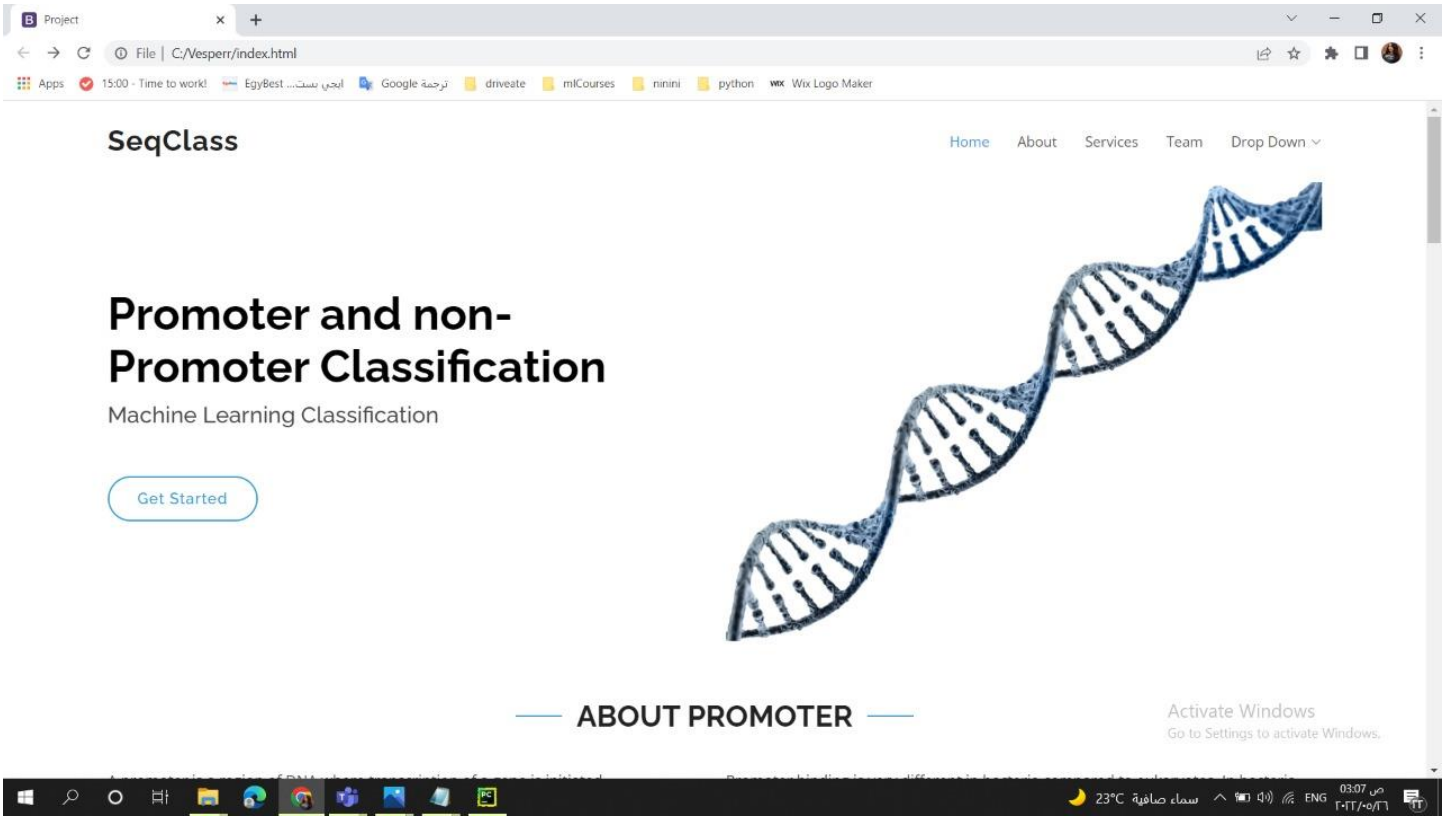
```
ppn = Perceptron(eta0=.01, max_iter=75)
ppn.fit(X_train, y_train)

print("Test = " ,accuracy_score(ppn.predict(X_test), y_test))
print("Train = " ,accuracy_score(ppn.predict(X_train), y_train))
```

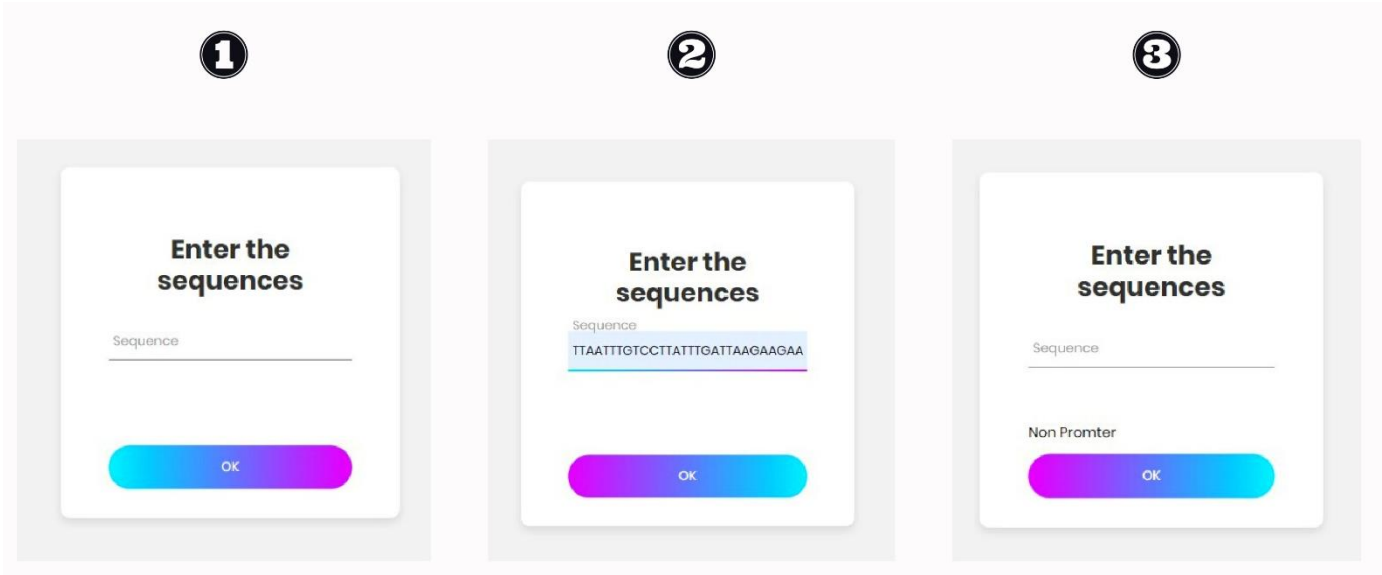
```
Test =  0.7331858407079646
Train =  0.9945790463546853
```

The main result: is how accurate the perceptron model _after training_ in classifying DNA sequences into a promoter and non-Promoter region
-The model gave the highest accuracy on train data
and 73.33% on test data

Final product :

A website linked with the perceptron model to predict whether any given DNA sequence is a promoter or non promoter





5.Conclusions:

- to further study and analyze promoter and non-promoter regions in order to treat certain genetic diseases there needed to be an optimized method that is able to classify them with high accuracy
- that's way we build a perceptron model to do just that

and it resulted in 73.33 % accuracy score and a time complexity of o(t) as it's a liner classifier

- then we proceeded to build a full functioning responsive website
- that can predict whether the inputted sequence is a promoter or non-promoter
- as it is directly linked with the model via flask micro web framework

6.References:

1. Sharan, Roded (4 January 2007). "Analysis of Biological Networks: Transcriptional Networks – Promoter Sequence Analysis" (PDF). Tel Aviv University. Retrieved 30 December 2012.
2. Asgari E., Mofrad M. R. K. (2015). Continuous distributed representation of biological sequences for deep proteomics and genomics. PLoS ONE 10: e0141287. 10.1371/journal.pone.0141287
3. NCBI, article: PMC6848157
4. www.genome.gov/genetics-glossary/Promoter
5. Copland JA, Sheffield-Moore M, Koldzic-Zivanovic N, Gentry S, Lamprou G, Tzortzatou-Stathopoulou F, Zoumpourlis V, Urban RJ, Vlahopoulos SA (June 2009). "Sex steroid receptors in skeletal differentiation and epithelial neoplasia is tissue-specific intervention possible"
6. Hobbs K, Negri J, Klinnert M, Rosenwasser LJ, Borish L (December 1998). "Interleukin-10 and transforming growth factor-beta promoter polymorphisms in allergies and asthma". American Journal of Respiratory and Critical Care Medicine.
7. Burchard EG, Silverman EK, Rosenwasser LJ, Borish L, Yandava C, Pillari A, Weiss ST, Hasday J, Lilly CM, Ford JG, Drazen JM (September 1999). "Association between a sequence variant in the IL-4 gene promoter and

FEV (1) in asthma". American Journal of Respiratory and Critical Care Medicine
8. www.frontiersin.org/articles/10.3389/fbioe.2019.00305/full
9. Kulozik AE, Bellan-Koch A, Bail S, Kohne E, Kleihauer E (May 1991). "Thalassemia intermedia: moderate reduction of beta globin gene transcriptional activity by a novel mutation of the proximal CACCC promoter element"
10. Petrij F, Giles RH, Dauwerse HG, Saris JJ, Hennekam RC, Masuno M, Tommerup N, van Ommen GJ, Goodman RH, Peters DJ (July 1995). "Rubinstein-Taybi syndrome caused by mutations in the transcriptional co-activator CBP". Nature. 376 (6538): 348–51
11. www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron
12. www.javatpoint.com/perceptron-in-machine-learning
13. www.code4example.com/pseudocode/pseudocode-examples/

7.Appendix A :

Source code:
https://colab.research.google.com/drive/1fGTjUtyMEY3jWmSJCMxp4wjXcHWT0xg?usp=sharing

Website (local hosting):
https://drive.google.com/drive/folders/1hwFEjyQf8Rt3FFJeaserfLvlRcg8Vnp9?usp=sharing