Sayed Mohammad Fatemi

student number: 40127793

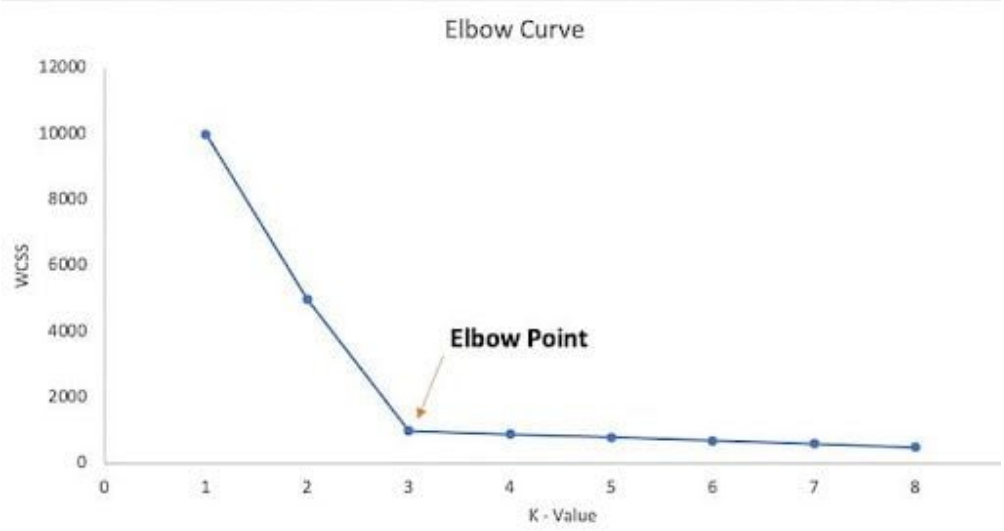
signal project

phase 2

در الگوریتم های مشین لرنینگ به طور کلی ما هایپر پارامتر هایی داریم که باید آن ها را بنابر اطلاعاتی که از دیتای خود داریم و سعی و خطا و نیز متد های موجود پارامتر ها را پیدا کنیم که مثلا برای الگوریتم ذکر شده در فاز ۲ مقدار "کا" را باید همین کار را کنیم

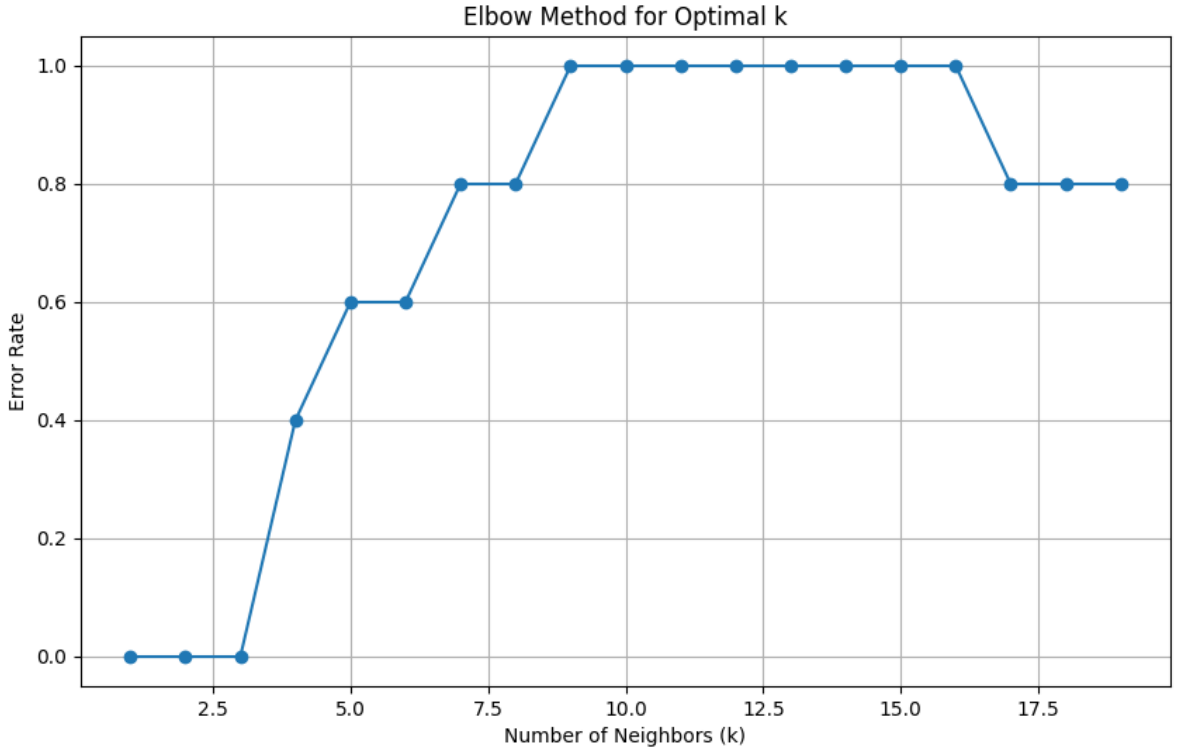
یکی از روش های موجود به شرح زیر است :

elbow method



درواقع در این متد باید مقدار های مختلف را تست کنیم و به یه جایی میرسیم که تابع خطای ما ناگهان به شدت افت میکند و این نقطه معمولا همان نقطه ی بهینه برای هایپرپارامتر ماست

طبق نموداری که برای پروژه ی خودمان رسم کردم به شکل زیر است



همانطور که میبینیم نمودار نرمی ندارد و چون تعداد داده ها کم است و نیز به علت نوع داده ها و برچسب های آن ها نمودار نرمی نداریم

همچنین بنظر میرسد که چون خطا به ازای ۱ تا ۳ همسایه صفر است دچار اورفیت شویم اگر بخواهیم این مقدار را اضافه کنیم ولی چون به ازای بقیه مقادیر اولا ارور در حال صعود است دوما جایی که ارور دوباره کم میشود یعنی تقریبا ۱۷ همچنان ارور خیلی بالاست فلذا مقدار ۳ همسایه در این مسیله و با این داده ها بنظر بهینه می آید

حالا کد را برایتان توضیح میدهیم

سورس کد :

ابتدا کتاب هاخانه های مورد نیاز برای رسم و مشین لرنینگ و داده و ... را وارد برنامه میکنم

```python
import os
from pydub import AudioSegment
import numpy as np
from scipy.fft import fft, fftfreq
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

متدی که توضیح دادم را در این تابع پیاده سازی کردم و دیتای ترین را به دو دسته ی ترین و ولیدشین تقسیم کردم چون در کاربردهای مشین لرنینگ جز دیتای تست و ترین یک دسته دیتا برای اعتبار سنجی هم داریم که هنگام ترین کردن نیز خطای مدل را با آن نیز میسنجیم سپس یک حلقه زدم و به ازای مقادیر مختلف تعداد همسایگی ها را تست کردم و مقدار خطای آن ها را بر حسب عملکرد مدل حساب کردم و نگه داشتم و در آخر نمودار خطا را بر حسب مقادیر مختلف همسایگی حساب کردم و در آخر مقدار مینیمم خطا را در آخرین ایندکسی که اتفاق افتاد چاپ کردم

```python
def elbow_method():
    X_train = []
```

```
    y_train = []

    for note_name, frequencies in known_notes.items():
        for freq in frequencies:
            X_train.append(freq)
            y_train.append(note_name)

    X_train = np.array(X_train).reshape(-1, 1)
    y_train = np.array(y_train)

    X_train_split, X_val, y_train_split, y_val =
    train_test_split(X_train, y_train, test_size=0.2, random_state=42)
    error_rates = []
    k_values = range(1, 20)

    for k in k_values:
        knn = KNeighborsClassifier(n_neighbors=k)
        knn.fit(X_train_split, y_train_split)

        y_pred = knn.predict(X_val)

        error_rate = 1 - accuracy_score(y_val, y_pred)
        error_rates.append(error_rate)

    plt.figure(figsize=(10, 6))
    plt.plot(k_values, error_rates, marker="o")
    plt.title("Elbow Method for Optimal k")
    plt.xlabel("Number of Neighbors (k)")
    plt.ylabel("Error Rate")
    plt.grid()
    plt.show()

    min_value = np.argmin(error_rates)

    indices = np.where(error_rates == min_value)[0]
    last_occurrence_index = indices[-1]
    best_k = k_values[last_occurrence_index]
    print(f"Best k value: {best_k}")
```

در این قسمت مانند فاز اول ابتدا مقدار فرکانسی که دامنه بیشتری در سیگنال دارد را جدا کرده ام به این صورت که بعد از نرمال سازی سیگنال و تبدیل فوریه و جدا کردن فرکانس های مثبت چون سیگنال حقیقی است مقدار ماکسیمم را خروجی داده ام

```python
def get_dominant_frequency(file_path):
    audio = AudioSegment.from_file(file_path, format="m4a")
    samples = np.array(audio.get_array_of_samples())

    sample_rate = audio.frame_rate

    samples = samples / np.max(np.abs(samples))

    n = len(samples)
    fft_values = fft(samples)
    frequencies = fftfreq(n, 1 / sample_rate)
```

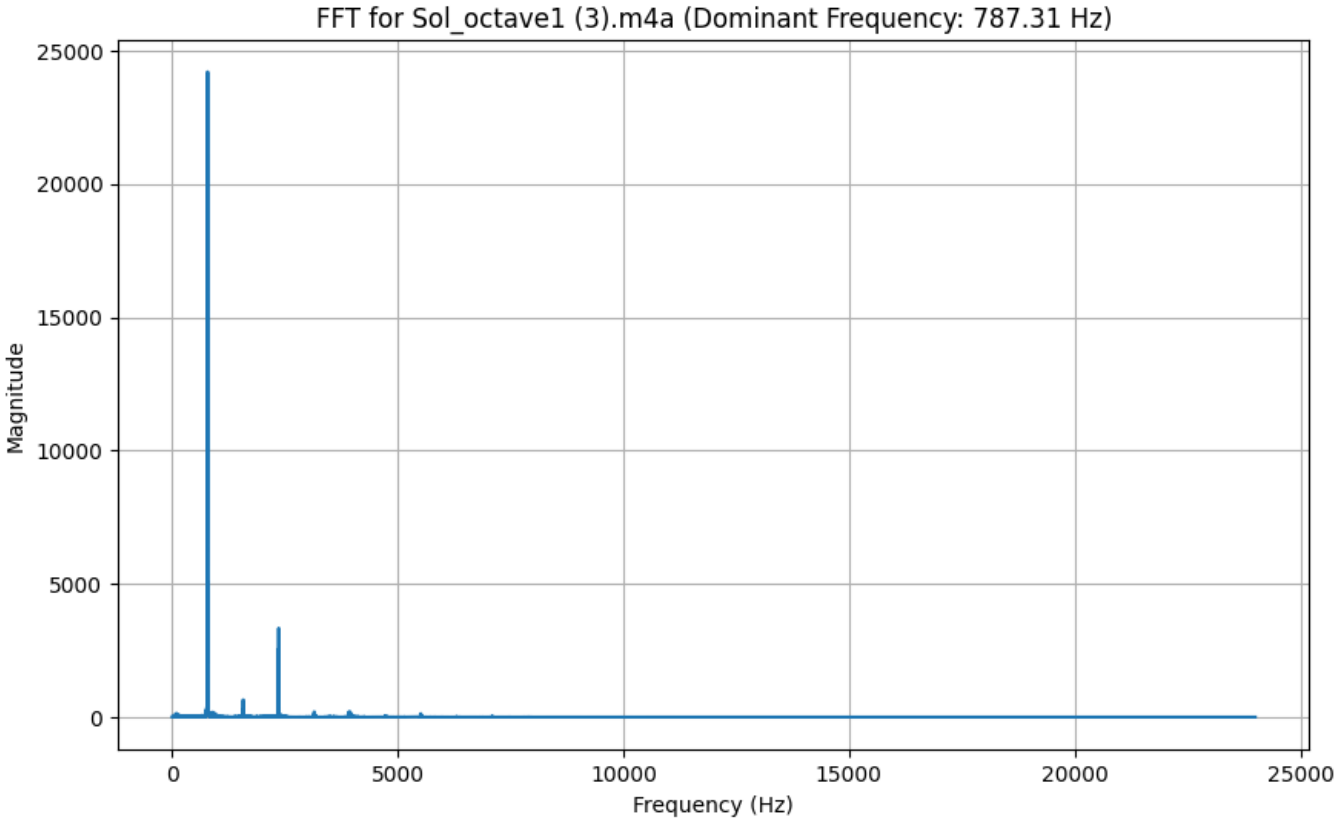
```
    positive_freq = frequencies[:n // 2]
    positive_fft_values = np.abs(fft_values[:n // 2])

    dominant_frequency = positive_freq[np.argmax(positive_fft_values)]

    return dominant_frequency, positive_freq, positive_fft_values
```

در این قسمت فایل های برچسب دار را خوانده ام و بر اساس نام و فرکانس آن ها جدا کرده ام و بعد از اینکه مدل را تنظیم کردم که بر اساس مقدار کای ۳ کلاس بندی کند و سپس روی دیتا ترین کردم و سپس به طریق مشابه فایل های تست را خواندم و با مدلی که ساخته بودم پیش بینی کردم که برچسب آن ها چه خواهد بود و در ضمن برای هر داده ی ترین نیز نمودار فرکانسی رسم میکنم و پیک آن را در تایتل نمایش می دهم

```python
known_notes_folder = "knn/train"
known_notes = {}

for filename in os.listdir(known_notes_folder):
    if filename.endswith(".m4a"):
        note_name = filename.split("_")[0]

        file_path = os.path.join(known_notes_folder, filename)
        dominant_frequency, positive_freq, positive_fft_values = get_dominant_frequency
(file_path)

        if note_name not in known_notes:
            known_notes[note_name] = []
        known_notes[note_name].append(dominant_frequency)

        # Plot FFT for each training file
        plt.figure(figsize=(10, 6))
        plt.plot(positive_freq, positive_fft_values)
        plt.title(f"FFT for {filename} (Dominant Frequency: {dominant_frequency:.2f} H
z)")
        plt.xlabel("Frequency (Hz)")
        plt.ylabel("Magnitude")
        plt.grid()
        plt.show()

X_train = []
y_train = []

for note_name, frequencies in known_notes.items():
    for freq in frequencies:
        X_train.append(freq)
        y_train.append(note_name)

X_train = np.array(X_train).reshape(-1, 1)
y_train = np.array(y_train)

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

```
unknown_notes_folder = "knn/test"

for filename in os.listdir(unknown_notes_folder):
    if filename.endswith(".m4a"):
        file_path = os.path.join(unknown_notes_folder, filename)
        dominant_frequency, _, _ = get_dominant_frequency(file_path)

        predicted_note = knn.predict([[dominant_frequency]])

        print(f"File: {filename}")
        print(f"Dominant Frequency: {dominant_frequency:.2f} Hz")
        print(f"Predicted Note: {predicted_note[0]}")
        print("-" * 40)

elbow_method()
```

برای مثال برای یکی از داده ها نمودار به شکل زیر است



و نیز بخشی از پیشبینی مدل به شکل زیر میباشد

```
(myenv) lucimad@sprites:~/Downloads/proj/signals-and-machine-learning$ python3 phase2.py
File: 11.m4a
Dominant Frequency: 991.98 Hz
Predicted Note: Si
--------------------------------------
File: 15.m4a
Dominant Frequency: 580.60 Hz
Predicted Note: Re
--------------------------------------
File: 12.m4a
Dominant Frequency: 519.53 Hz
Predicted Note: Do
--------------------------------------
File: 10.m4a
Dominant Frequency: 660.39 Hz
Predicted Note: Mi
--------------------------------------
File: 1.m4a
Dominant Frequency: 996.02 Hz
Predicted Note: Si
--------------------------------------
File: 5.m4a
Dominant Frequency: 788.59 Hz
Predicted Note: Sol
--------------------------------------
File: 3.m4a
Dominant Frequency: 1048.66 Hz
Predicted Note: Do
```

در این قسمت برای داده های ترین و تست نمودار میله ای رسم کردم

```python
# bar plot for training data frequencies
plt.figure(figsize=(12, 6))
for note_name, frequencies in known_notes.items():
    plt.bar(note_name, np.mean(frequencies), label=note_name)
plt.title("Training Data Frequencies")
plt.xlabel("Note")
plt.ylabel("Frequency (Hz)")
plt.legend()
plt.show()

# bar plot for test data frequencies
test_frequencies = []
test_notes = []

for filename in os.listdir(unknown_notes_folder):
    if filename.endswith(".m4a"):
        file_path = os.path.join(unknown_notes_folder, filename)
        dominant_frequency, _, _ = get_dominant_frequency(file_path)
        predicted_note = knn.predict([[dominant_frequency]])[0]
        test_frequencies.append(dominant_frequency)
        test_notes.append(predicted_note)

plt.figure(figsize=(12, 6))
plt.bar(test_notes, test_frequencies)
plt.title("Test Data Frequencies")
```

```
plt.xlabel("Predicted Note")
plt.ylabel("Frequency (Hz)")
plt.show()
```