



**Universidad Nacional de San
Agustín de Arequipa**

**Escuela Profesional de Ciencia de la
Computación**

**Computación Molecular Biológica
(Código: 1005155)**

Semestre 2020A

Indice

1	Curriculum Vitae en Formato ICACIT	4
2	Sílabo del Curso en Formato DUFA	7
3	Sílabo del Curso en Formato ICACIT	12
4	Prueba de Entrada	15
4.1	Evidencias	18
5	Evaluación Primer Parcial	19
5.1	Evidencias	21
6	Evaluación Segundo Parcial	22
6.1	Evidencias	29
7	Evaluación Tercer Parcial	30
7.1	Evidencias	64
8	Evaluación Continua 1	65
8.1	Evidencias	70
9	Evaluación Continua 2	71
9.1	Evidencias	83

10	Evaluación Continua 3	84
10.1	Evidencias	89
11	Registro de Asistencia	90
12	Registro de Notas	92
13	Material del Curso	94
14	Evaluación de desempeño	96
15	Informe Final del Curso	101



1. Curriculum Vitae en Formato ICACIT



UNIVERSIDAD NACIONAL DE SAN AGUSTÍN
Facultad de INGENIERÍA DE PRODUCCIÓN Y SERVICIOS
Escuela Profesional de Ingeniería DE SISTEMAS

Curriculum Vitae del Docente

Nombre	VICENTE ENRIQUE MACHACA ARCEDA
Educación	<ul style="list-style-type: none">· Magíster en ciencias informática con mención en tecnologías de información, 2016 / Universidad Nacional de San Agustín· Ingeniero de sistemas, 2017 / Universidad Nacional de San Agustín· Bachiller en Ingeniería de sistemas, 2011 / Universidad Nacional de San Agustín
Experiencia Académica	<ul style="list-style-type: none">· Cátedra de Postgrado; Universidad Universidad Nacional de San Agustín; 2017, 2018· Cátedra de Pregrado; Universidad Universidad Nacional de San Agustín; 2017, 2018· Cátedra de Pregrado; Universidad La Salle; 2017, 2018
Experiencia No Académica	<ul style="list-style-type: none">· Vex Soluciones E.I.R.L.; Analista; 2017 - 2017.· Tata Consultancy Services – Tcs; System Engennier; 2013 - 2014.· Superintendencia Nacional de Aduanas y Administración Tributaria (SUNAT); IPM: 2011 – 2014.· Coriing Eirl; Analista; 2011 – 2013· Ctd V&C Sac; Analista; 2010 – 2011· Regesa Scrl; Programador; 2009 – 2010
Registro Profesional	Colegio de Ingenieros del Perú, CIP: 211444
Membresía actual en organizaciones profesionales	<ul style="list-style-type: none">· Colegio de Ingenieros del Perú, 2018. CIP: 211444

Honores y Premios	<ul style="list-style-type: none"> · Acreedor a una beca integral para estudiar una maestría en ciencias informática. · Alumno revelación de la maestría en ciencias informática
Actividades de Servicio (dentro y fuera de la Institución)	
Publicaciones y Presentaciones (últimos 5 años)	<p>Publicaciones – Artículos</p> <ul style="list-style-type: none"> · Small Ship Detection on Optical Satellite Imagery with YOLO and YOLT. Presentado en: FTC 2020 - Future of Information and Communication Conference San Francisco, EEUU, 2020. · Fast Car Crash Detection. Presentado en: CLEI 2018 - The Latin American Computing Conference, São Paulo, Brasil, 2018. · Fast Face Detection in Violent Video Scenes. Publicado en: ScienceDirect. Presentado en: CLEI 2016 - The Latin American Computing Conference, Valparaiso, Chile, 2016. · Real Time Violence Detection in Video with ViF and Horn-Schunck. Publicado en: LACCEI. Presentado en: The Latin American and Caribbean Consortium of Engineering Institutions, San Jose, Costa Rica, 2016. · Optimization model for face detection in video sequences. Publicado en: LACCEI. Presentado en: The Latin American and Caribbean Consortium of Engineering Institutions, San Jose, Costa Rica, 2016. · Real Time Violence Detection in Video. Publicado en: IEEE Explore y IET Digital Library. Presentado en: International Conference on Pattern Recognition Systems, Talca, Chile, 2015.
Actividades de Desarrollo Profesional (últimos 3 años)	<ul style="list-style-type: none"> · Docente con experiencia en los niveles de pregrado y postgrado en diversas la Universidad Nacional de San Agustín y la Universidad La Salle. · Ponencias en Conferencias Internacionales LACCEI 2018, CLEI 2016.



2. Sílabo del Curso en Formato DUFA

UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA



VICERRECTORADO ACADÉMICO

FACULTAD DE INGENIERIA DE PRODUCCION Y SERVICIOS

DEPARTAMENTO ACADÉMICO DE INGENIERIA DE SISTEMAS E INFORMATICA

SÍLABO 2020 - A

ASIGNATURA: COMPUTACION MOLECULAR BIOLOGICA (E)

1. INFORMACIÓN ACADÉMICA

Periodo académico:	2020 - A										
Escuela Profesional:	CIENCIA DE LA COMPUTACIÓN										
Código de la asignatura:	1005155										
Nombre de la asignatura:	COMPUTACION MOLECULAR BIOLOGICA (E)										
Semestre:	IX (noveno)										
Duración:	17 semanas										
Número de horas (Semestral)	<table border="1"><tr><td>Teóricas:</td><td>2.0</td></tr><tr><td>Prácticas:</td><td>2.0</td></tr><tr><td>Seminarios:</td><td>0.0</td></tr><tr><td>Laboratorio:</td><td>0.0</td></tr><tr><td>Teórico-prácticas:</td><td>2.0</td></tr></table>	Teóricas:	2.0	Prácticas:	2.0	Seminarios:	0.0	Laboratorio:	0.0	Teórico-prácticas:	2.0
Teóricas:	2.0										
Prácticas:	2.0										
Seminarios:	0.0										
Laboratorio:	0.0										
Teórico-prácticas:	2.0										
Número de créditos:	4										
Prerrequisitos:	ESTRUCTURA DE DATOS AVANZADAS (1003233)										

2. INFORMACIÓN DEL DOCENTE, INSTRUCTOR, COORDINADOR

DOCENTE	GRADO ACADÉMICO	DPTO. ACADÉMICO	HORAS	HORARIO
MACHACA ARCEDA, VICENTE	Magister	INGENIERIA DE SISTEMAS E INFORMATICA	0	Mar: 17:40-20:10

3. INFORMACIÓN ESPECIFICA DEL CURSO (FUNDAMENTACIÓN, JUSTIFICACIÓN)

El uso de métodos computacionales en las ciencias biológicas se ha convertido en una de las herramientas claves para el campo de la biología molecular, siendo parte fundamental en las investigaciones de esta área. En Biología Molecular, existen diversas aplicaciones que involucran tanto al ADN, al análisis de proteínas o al secuenciamiento del genoma humano, que dependen de métodos computacionales. Muchos de estos problemas son realmente complejos y tratan con grandes conjuntos de datos. Este curso puede ser aprovechado para ver casos de uso concretos de varias áreas de

conocimiento de Ciencia de la Computación como: Lenguajes de Programación (PL), Algoritmos y Complejidad (AL), Probabilidades y Estadística, Manejo de Información (IM), Sistemas Inteligentes (IS).

4. COMPETENCIAS/OBJETIVOS DE LA ASIGNATURA

La comprensión intelectual y la capacidad de aplicar las bases matemáticas y la teoría de la informática (Resultado [a] nivel 2).

Analiza, diseña y propone soluciones frente a problemas bioinformáticos. (Resultado [b] nivel 1, Resultado [c] nivel 1, Resultado [d] nivel 1).

Sabe cómo utilizar y conoce las bases computacionales de herramientas modernas de secuenciamiento, alineamiento, árboles filogenéticos y mapeo de genomas. (Resultado [a] nivel 2 y Resultado [h] nivel 2).

5. CONTENIDO TEMATICO

PRIMERA UNIDAD

Capítulo I: Biología molecular

Tema 01: Introducción

Tema 02: Biología de la célula

Tema 03: Bases de datos en bioinformática

Tema 04: De DNA a proteínas

Tema 05: Estructura del DNA y replicación de DNA

Tema 06: Secuenciamiento de DNA

SEGUNDA UNIDAD

Capítulo II: Alineamiento de Secuencias

Tema 07: Alineamiento de secuencias de DNA y aminoácidos

Tema 08: Dot matrix

Tema 09: Programación dinámica

Tema 10: BLAST

TERCERA UNIDAD

Capítulo III: Árboles Filogenéticos

Tema 11: Introducción y relaciones filogenéticas

Tema 12: UPGMA

Tema 13: Neighbor joining

Tema 14: Métodos basados en caracteres

CUARTA UNIDAD

Capítulo IV: Ensamblaje de Secuencias

Tema 15: El problema np-hard de ensamblaje de secuencias de ADN

Tema 16: Técnicas basadas en grafos y k-mer

Tema 17: Técnicas basadas en heurísticas

QUINTA UNIDAD

Capítulo V: Tópicos en Bioinformática

Tema 18: Métodos para la clasificación de nuevas cepas de virus

Tema 19: Detección temprana de enfermedades en base al análisis del genoma

Tema 20: Farmacogenética y predicción a la respuesta a medicamentos

6. PROGRAMACIÓN DE ACTIVIDADES DE INVESTIG. FORMATIVA Y RESPONSABILIDAD SOCIAL

6.1. Métodos

Expositivo en clases teóricas y desarrollo de un trabajo práctico.

6.2. Medios

Classroom, DUTIC, Google meet.

6.3. Formas de organización

Clases teóricas, exposición de clases magistrales.

6.4. Programación de actividades de investigación formativa y responsabilidad social

Desarrollo de talleres y exposición de entregables.

7. CRONOGRAMA ACADÉMICO

SEMANA	TEMA	DOCENTE	%	ACUM.
	Introducción	V. Machaca	5	5.00
	Biología de la célula	V. Machaca	5	10.00
	Bases de datos en bioinformática	V. Machaca	5	15.00
	De DNA a proteínas	V. Machaca	5	20.00
	Estructura del DNA y replicación de DNA	V. Machaca	5	25.00
	Secuenciamiento de DNA	V. Machaca	5	30.00
	Alineamiento de secuencias de DNA y aminoácidos	V. Machaca	5	35.00
	Dot matrix	V. Machaca	5	40.00
	Programación dinámica	V. Machaca	5	45.00
	BLAST	V. Machaca	5	50.00
	Introducción y relaciones filogenéticas	V. Machaca	5	55.00
	UPGMA	V. Machaca	5	60.00
	Neighbor joining	V. Machaca	5	65.00
	Métodos basados en caracteres	V. Machaca	5	70.00
	El problema np-hard de ensamblaje de secuencias de ADN	V. Machaca	5	75.00
	Técnicas basadas en grafos y k-mer	V. Machaca	5	80.00
	Técnicas basadas en heurísticas	V. Machaca	5	85.00
	Métodos para la clasificación de nuevas cepas de virus	V. Machaca	5	90.00
	Detección temprana de enfermedades en base al análisis del genoma	V. Machaca	5	95.00
	Farmacogenética y predicción a la respuesta a medicamentos			

8. ESTRATEGIAS DE EVALUACIÓN

8.1. Evaluación del aprendizaje

Evaluación Continua. Práctica y Laboratorios en cada clase sobre los temas realizados, tanto para el primer parcial (EC1), segundo parcial (EC2) y tercer parcial (EC3).

Evaluación Periódica. Al ser un curso basado en lenguajes de programación, la evaluación periódica consta en la revisión de un trabajo de implementación.

8.2. Cronograma de evaluación

EVALUACIÓN	FECHA DE EVALUACIÓN	EXAMEN TEORÍA	EVAL. CONTINUA	TOTAL (%)
Primera Evaluación Parcial		15%	15%	30%
Segunda Evaluación Parcial		15%	15%	30%
Tercera Evaluación Parcial		20%	20%	40%
TOTAL				100%

9. REQUISITOS DE APROBACIÓN DE LA ASIGNATURA

Para aprobar el curso se deberá haber presentado todos sus trabajos. Los trabajos o tareas deberán ser originales, la copia o plagio a cualquier tipo de nivel, o cualquier tipo de actitud deshonesta, será castigado con cero en todo el componente donde se haya detectado la copia.

10. BIBLIOGRAFIA: AUTOR, TÍTULO, AÑO, EDITORIAL

10.1. Bibliografía básica obligatoria

- [1] Aluru, S., editor (2006). Handbook of Computational Molecular Biology. Computer and Information Science Series. Chapman & Hall, CRC, Boca Raton, FL.
- [2] Archibald, John M. Genomics: A Very Short Introduction. Vol. 559. Oxford University Press, 2018.
- [3] Xiong, Jin. Essential bioinformatics. Cambridge University Press, 2006.

10.2. Bibliografía de consulta

- [4] Korpelainen E, Tuimala J, Somervuo P, Huss M, Wong G. RNA-seq data analysis: a practical approach. CRC press; 2014 Sep 19.
- [5] Kim IJ, editor. Cancer Genetics and Genomics for Personalized Medicine. CRC Press; 2017 Apr 11.

Arequipa, 08 de Junio del 2020

MACHACA ARCEDA, VICENTE



3. Sílabo del Curso en Formato ICACIT

Sílabos del Curso

ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

1. Nombre del curso:

Código	Nombre	Semestre
1005155	Computación molecular biológica	2020-A

2. Créditos y horas semanales:

Nº créditos	H. Teoría	H. Práctica	H. T-P	H. Lab	T. Horas
6	2	2	2		6

3. Nombre del instructor o coordinador del curso:

MSc. Vicente Machaca Arceda

4. Libro texto: Título, autor y año:

a. Obligatoria

Título	Autor	Año
Essential bioinformatics	Essential bioinformatics	2006

b. Otros materiales suplementarios

Título	Autor	Año
A Very Short Introduction	Archibald, John M	2018

5. Información específica del curso:

a. Breve descripción del contenido del curso:

El curso tiene como objetivo que el alumno tenga un conocimiento sólido de los problemas biológicos moleculares que desafía la computación y que el alumno sea capaz de abstraer la esencia de los diversos problemas biológicos para plantear soluciones usando sus conocimientos de Ciencia de la Computación.

b. Requisitos previos o correquisitos:

1703238 - Estructuras de datos avanzadas

c. Obligatorio o Electivo:

Obligatorio		Electivo	X
-------------	--	----------	---

6. Objetivos específicos del curso:

La comprensión intelectual y la capacidad de aplicar las bases matemáticas y la teoría de la informática (Resultado [a] nivel 2).

Analiza, diseña y propone soluciones frente a problemas bioinformáticos. (Resultado [b] nivel 1, Resultado [c] nivel 1, Resultado [d] nivel 1).

Sabe cómo utilizar y conoce las bases computacionales de herramientas modernas de secuenciamiento, alineamiento, árboles filogenéticos y mapeo de genomas. (Resultado [a] nivel 2 y Resultado [h] nivel 2).

7. Breve lista de temas a ser abordados en el curso:

Biología molecular
Alineamiento de Secuencias
Árboles Filogenéticos
Ensamblaje de Secuencias
Tópicos en Bioinformática



4. Prueba de Entrada

EXAMEN DE ENTRADA

Docente: MSc. Vicente Machaca Arceda
Abril, 2020

Apellidos:

Nombre:

CUI:

1. Explique qué entiende por ADN. **(4 puntos)**
2. Explique qué entiende por genes y proteínas. **(4 puntos)**
3. Explique que son las mutaciones y a que se debe su aparición. **(4 puntos)**
4. Explique que son los árboles filogenéticos. **(4 puntos)**
5. Implementar un programa en el lenguaje de su preferencia que reciba como entrada dos cadenas de texto y retorne un valor numérico indicando el grado de similitud entre dichas cadenas. Usted puede definir qué criterios tomar para retornar el grado de similitud. **(4 puntos)**

Ejemplos de cadenas similares:

cadena_1: “ACGT”

cadena_2: “ACGGT”

cadena_1: “GTAACGT”

cadena_2: “GTAAGT”

cadena_1: “ACGT”

cadena_2: “AGGT”

EXAMEN DE ENTRADA

Docente: MSc. Vicente Machaca Arceda
Abril, 2020

Apellidos:

Nombre:

CUI:

1. Explique qué entiende por ADN. **(4 puntos)**

Cadena de moléculas basadas en Adenine, Cytosine, Guanine y Tymine. Dicha cadena forma una doble helice, y son el sustrato para la creación de proteínas.

2. Explique qué entiende por genes y proteínas. **(4 puntos)**

Los genes son segmentos del ADN y cada gen forma entre una a más proteínas (isomorfos)

3. Explique que son las mutaciones y a que se debe su aparición. **(4 puntos)**

Las mutaciones son cambios en las bases nitrogenadas del ADN, estos cambios originan proteínas malformadas y funciones deterioradas. Algunas causas de su origen son los hábitos de fumar, exposición a radiación y algunas sustancias químicas.

4. Explique que son los árboles filogenéticos. **(4 puntos)**

Es un método gráfico de computación para mostrar el parentesco entre muestras según su similitud.

5. Implementar un programa en el lenguaje de su preferencia que reciba como entrada dos cadenas de texto y retorne un valor numérico indicando el grado de similitud entre dichas cadenas. Usted puede definir qué criterios tomar para retornar el grado de similitud. **(4 puntos)**

Ejemplos de cadenas similares:

cadena_1: "ACGT"
cadena_2: "ACGGT"

```
index = 0
similitud = 0
for c in cadena_1:
    if index < len(cadena_2) and c == cadena_2[index]:
        similitud += 1
    index += 1
return similitud
```

4.1 Evidencias

Rindieron la Prueba de Entrada 16 estudiantes de los 17 estudiantes matriculados, lo que representa un 94%.

Tabla 4.1: Notas y evidencias del examen de entrada

Apellidos y Nombres	P1	P2	P3	P4	P5	NOTA	EVIDENCIA
ARCOS/PONCE, SERGIO MANUEL	3	3	4	0	4	14	Link
BERMUDEZ/NAVARRO, WILLIAN	3	3	4	2	4	16	Link
CAYLLAHUE/CCORA, RENZO AUGUSTO	3	3	4	2	0	12	Link
CHAMBI APAZA, SYOMIRA INES	0	3	3	2	4	12	Link
CHAVEZ LOPEZ CAROLINA BONNIE	2	4	3	2	0	11	Link
CONDORI/MANSILLA, WILLIAM	2	2	0	2	4	10	Link
DEXTRE/AIQUIPA, MARKS CRISTOPHER						0	Link
DIAZ/VENTURA, CELSO EFRAIN NOEL	1	2	2	0	0	5	Link
GUARDIA/ZENTENO, IGOR ALFRED	1	3	3	0	4	11	Link
HUAYPUNA/HUANCA, JOHANN FRANZ	2	3	3	0	4	12	Link
LEON/PAREDES, GUSTAVO MARTIN	4	4	4	0	0	12	Link
SONCCO/LUPA, JEAN CARLOS	2	3	3	3	0	11	Link
TAMO/TURPO, ERIKA JUDITH	2	3	3	0	4	12	Link
TORRES/LIMA, JOSE MANUEL	2	2	2	0	4	10	Link
VILLANUEVA/SANCHEZ, FERNANDO	3	3	3	2	0	11	Link
VISA/FLORES, ALBERTO	2	3	0	3	4	12	Link



5. Evaluación Primer Parcial

Primer examen parcial

MSc. Vicente Machaca Arceda

8 de junio de 2020

DOCENTE	CARRERA	CURSO
MSc. Vicente Machaca Arceda	Escuela Profesional de Ciencia de la Computación	Computación Molecular Biológica

1. Preguntas

1. ¿En que lugar de las celulas, NO esta presente el DNA?
2. ¿Cuáles son las bases nitrogenadas presentes en el RNA?
3. ¿Cuáles son las bases nitrogenadas presentes en el DNA?
4. What is bioinformatics?
5. ¿Qué carbono del azucar ribosa es utilizado para la unión de los nucleotidos durante la replicación de DNA?
6. ¿Qué son los "codons" durante la transcripción?
7. ¿Qué es un dNTP?
8. ¿Qué es un ddNTP?
9. ¿Cuáles son correctas respecto a gel electrophoresis y capillary electrophoresis
10. ¿Cuál es la característica del cDNA durante RNA sequencing?

Apellidos y Nombres	EP2	Evidencia
ARCOS/PONCE, SERGIO MANUEL	16	Enlace
BERMUDEZ/NAVARRO, WILLIAN BRAULIO	0	Enlace
CAYLLAHUE/CCORA, RENZO AUGUSTO	12	Enlace
CHAMBI APAZA, SYOMIRA INES	13	Enlace
CHAVEZ LOPEZ CAROLINA BONNIE	10	Enlace
CONDORI/MANSILLA, WILLIAM SILVERIO	10	Enlace
DEXTRE/AIQUIPA, MARKS CRISTOPHER	11	Enlace
DIAZ/VENTURA, CELSO EFRAIN NOEL	6	Enlace
GUARDIA/ZENTENO, IGOR ALFRED	14	Enlace
HUAYPUNA/HUANCA, JOHANN FRANZ	7	Enlace
HUISA QUISPE, JOSE LUIS	7	Enlace
LEON/PAREDES, GUSTAVO MARTIN	12	Enlace
SONCCO/LUPA, JEAN CARLOS	6	Enlace
TAMO/TURPO, ERIKA JUDITH	6	Enlace
TORRES/LIMA, JOSE MANUEL	18	Enlace
VILLANUEVA/SANCHEZ, FERNANDO THOMAS	12	Enlace
VISA/FLORES, ALBERTO	14	Enlace

5.1 Evidencias

Rindieron la Segunda Evaluación Parcial 17 estudiantes de los 17 estudiantes matriculados, lo que representa un 100.0%.

La nota promedio de los estudiantes que rindieron la Segunda Evaluación Parcial es 11 puntos. Las notas y la evidencia de la Segunda Evaluación Parcial se encuentran en la siguiente tabla:



6. Evaluación Segundo Parcial

Segundo examen parcial

MSc. Vicente Machaca Arceda

4 de julio de 2020

DOCENTE	CARRERA	CURSO
MSc. Vicente Machaca Arceda	Escuela Profesional de Ciencia de la Computación	Computación Molecular Biológica

1. Preguntas

1. Explique los pasos a seguir en el secuenciamiento de Sanger (max. 50 palabras) (3pt).
2. Explique a que hace referencia el termino *Alternative Splicing* (3pt).
3. Explique cual es la causa de que solo el 1% del genoma humano se sintetice a proteinas (3pt).
4. Encuentre el mejor alineamiento global entre las secuencias **ATCG** y **TTCG**, con el siguiente *scoring scheme*: +1 for match, -1 for mismatch and -1 for an alignment with a gap.
5. Encuentre el mejor alineamiento local entre las secuencias **ATACTGGG** y **TGACTGAG**, con el siguiente *scoring scheme*: +1 for match, -1 for mismatch and -1 for an alignment with a gap.

UNIVERSIDAD NACIONAL DE SAN AGUSTÍN
FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS
ESCUELA PROFESIONAL CIENCIA DE LA COMPUTACIÓN



COMPUTACIÓN MOLECULAR BIOLÓGICA

Examen

ALUMNO:

TORRES LIMA, JOSE

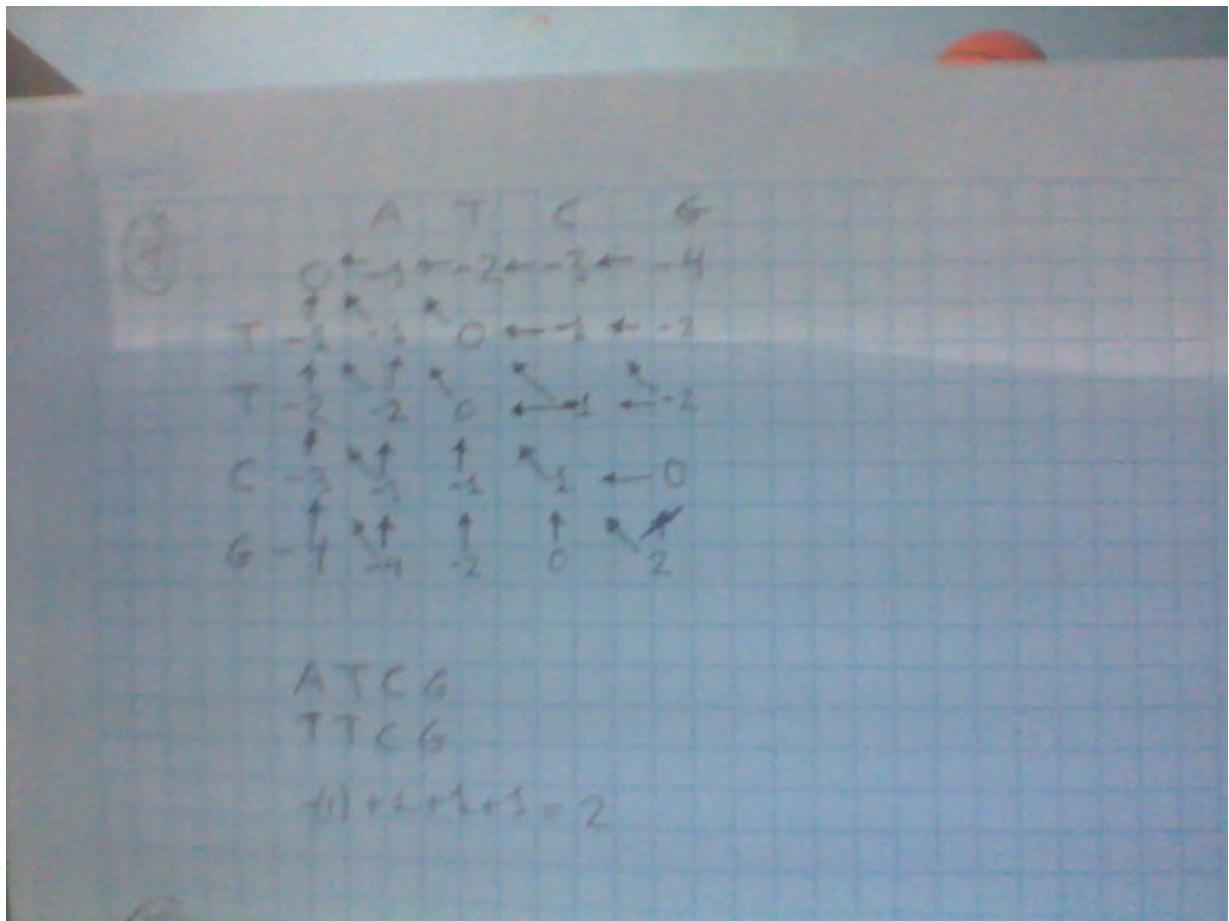
DOCENTE:

Prof: MSc. VICENTE MACHACA ARCEDA

AREQUIPA - PERÚ

2020

- Explique los pasos a seguir en el secuenciamiento de Sanger (max. 50 palabras) (3pt)
- Explique a que hace referencia el termino Alternative Splicing (3pt).
Permite obtener a partir de un transcripto primario de ARNm o pre-ARNm distintas isoformas de ARNm y proteínas, las cuales pueden tener funciones diferentes y a menudo opuestas. Combinando los exones. Se generan proteínas. proteinas isomorfas a partir del ARN primario desde el ADN
- Explique cual es la causa de que solo el 1 % del genoma humano se sintetice a proteinas (3pt)
Debido a los exones que son el 1% y los intrones el 99%
- Encuentre el mejor alineamiento global entre las secuencias ATCG y TTCTG, con el siguiente scoring scheme: +1 for match, -1 for mismatch and -1 for an alignment with a gap.



- Encuentre el mejor alineamiento local entre las secuencias ATACTGGG y TGACTGAG,, con el siguiente scoring scheme: +1 for match, -1 for mismatch and -1 for an alignment with a gap.

(5)

T G A C T G A G
d d d d d d d d
A d d d d d d d d
T d d d d d d d d
A d d d d d d d d
C d d d d d d d d
T d d d d d d d d
G d d d d d d d d
G d d d d d d d d
G d d d d d d d d

ACTGGG

ACTGAG

$$1+1+3+1-1+3=9$$

ACTG

ACTG

$$(1+1)+(1+3)=6$$

1.-

Se debe dividir el ADN con algún método, por inducción de calor por ejemplo, con esto el adn se divide en fragmentos pequeños.

Se añade dntps fluorescentes, para que sirva de marcador y podamos saber con que marcas terminan los fragmentos.

Desde aquí con ayuda de un aparato con carga se podrá alinear los fragmentos para poder estirarlos y ver las secuencias de los fragmentos e identificar los marcadores fluorescentes

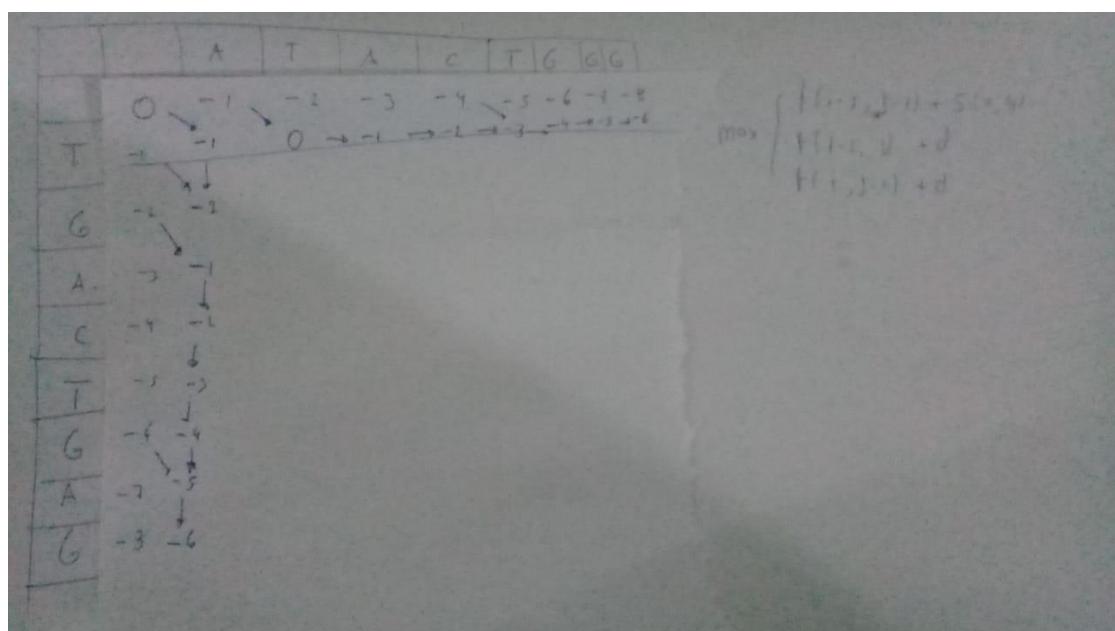
2.-

A las distintas formas que puede tomar un arn o proteínas, dando como resultado muchas veces mutaciones.

3.-

Porque posee 3.2 millones de bp

4.-



		A	T	C	G
	T	0	0	0	0
T	0	0	1	0	0
C	0	0	0	2	0
G	0	0	0	1	3

$$M_{\max} \begin{cases} f(i-i, j-1) + S(x_i, g_j) \\ f(i-j, j) + d \\ f(i, j-1) + d \\ 0 \end{cases}$$

$$\frac{T}{T} \frac{C}{C} \frac{G}{G} = 3$$

$$\frac{\bar{T}}{\bar{T}} \frac{C}{C} = 2$$

$$\frac{C}{C} \frac{G}{G} = 1$$

6.1 Evidencias

Rindieron la Primera Evaluación Parcial 16 estudiantes de los 17 estudiantes matriculados, lo que representa un 100.0%.

La nota promedio de los estudiantes que rindieron la Primera Evaluación Parcial es 10 puntos. Las notas y la evidencia de la Primera Evaluación Parcial se encuentran en la siguiente tabla:

Apellidos y Nombres	Nota	Evidencia
Amable Romero, Diego Javier	18.0	Link
Bernal Chahuayo, Luis Antonio	14.0	Link
Caceres Zegarra, Luis Gustavo	15.5	Link
Espinel Quispe, Ingrid Sally	04.5	Link
Gordillo Viña, Karen	15.0	Link
Gutierrez Salazar, Enrique Alonzo	18.7	Link
Hancco Tancayllo, Hermith	13.7	Link
Huaman Canqui Jair, Francesco	15.5	Link
Lacuaña Apaza, Margarita	10.3	Link
Larraondo Lancho, Alejandro Jesús	18.7	Link
Mamani Chirinos, Luis	05.7	Link
Mendoza Villarroel, Alexis	19.0	Link
Quincho Mamani, Lehi	09.2	Link
Quispe Quicano, Julio Cesar	11.7	Link
Turpo Apaza, Crhristian Andrew	17.7	Link
Uñapilco Chambi, Katherin	15.7	Link

Tabla 6.1: Resultados Segunda Evaluación Parcial



7. Evaluación Tercer Parcial

Tercer examen parcial

MSc. Vicente Machaca Arceda

30 de julio de 2020

DOCENTE	CARRERA	CURSO
MSc. Vicente Machaca Arceda	Escuela Profesional de Ciencia de la Computación	Computación Molecular Biológica

1. Competencias del curso

- Aplica las bases matemáticas y la teoría de la informática en algoritmos de Bioinformática.
- Analiza, diseña y propone soluciones frente a problemas bioinformáticos.
- Sabe cómo utilizar y conoce las bases computacionales de herramientas modernas de secuenciamiento, alineamiento, árboles filogenéticos y mapeo de genomas.

2. Competencias del trabajo

- Implementar un paper de investigación en Bioinformática.

3. Equipos y materiales

- Editor de texto Latex

4. Entregables

- Se debe elaborar un informe en Latex donde se desarrolle el trabajo solicitado.
- El informe se desarrollará en grupos de 4.
- El informe deberá estar correctamente citado utilizando las normas APA o IEEE.

5. Descripción del trabajo

Implementar un artículo académico de Bioinformática. Estos trabajos han sido escogidos, de manera tal que tengan una complejidad de nivel medio, se cuente con todos los recursos (bases de datos, librerías) y pueden ser llevados a cabo en una PC básica. Los trabajos a implementar son:

- *DNA sequence similarity analysis using image texture analysis based on first-order statistics* [1].
- *Use of image texture analysis to find DNA sequence similarities* [2].
- A New Local Search Algorithm for the DNA Fragment Assembly Problem [3].
- *Toward an Alignment-Free Method for Feature Extraction and Accurate Classification of Viral Sequences* [4].

6. Rúbricas

Rúbrica	Cumple	Cumple con obs.	No cumple
Informe: El informe debe estar en Latex, con un formato limpio y fácil de leer. Además, debe contener: descripción de los algoritmos utilizados, código fuente y resultados obtenidos.	3	1.5	0
Implementación: Los alumnos han logrado implementar el algoritmo del paper.	7	3.5	0
Resultados: El grupo ha logrado replicar los resultados en un 50 %. Es decir, no es necesario utilizar todas las bases de datos utilizadas en el paper.	5	2.5	0
Presentación: El alumno demuestra dominio del tema y conoce con exactitud cada parte de su código. Además, demuestra conocer la base matemática de su implementación. Se realizará preguntas a cada integrante del grupo, si un alumno no logra responder, tendrá cero en esta rúbrica.	5	2.5	0

Referencias

- [1] E. Delibaş and A. Arslan, “Dna sequence similarity analysis using image texture analysis based on first-order statistics,” *Journal of Molecular Graphics and Modelling*, p. 107603, 2020.
- [2] W. Chen, B. Liao, and W. Li, “Use of image texture analysis to find dna sequence similarities,” *Journal of theoretical biology*, vol. 455, pp. 1–6, 2018.
- [3] E. Alba and G. Luque, “A new local search algorithm for the dna fragment assembly problem,” in *European Conference on Evolutionary Computation in Combinatorial Optimization*, pp. 1–12, Springer, 2007.
- [4] D. Lebatteux, A. M. Remita, and A. B. Diallo, “Toward an alignment-free method for feature extraction and accurate classification of viral sequences,” *Journal of Computational Biology*, vol. 26, no. 6, pp. 519–535, 2019.

UNIVERSIDAD NACIONAL DE SAN AGUSTÍN
FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS
ESCUELA PROFESIONAL CIENCIA DE LA COMPUTACIÓN



COMPUTACIÓN MOLECULAR BIOLÓGICA

Informe Toward and alignment-free method

ALUMNOS:

TORRES LIMA, JOSE
DIAZ VENTURA, NOEL
HUAYPUNA HUANCA, JOHANN
HUISA QUISPE, LUIS

DOCENTE:

Prof: MSc. VICENTE MACHACA ARCEDA

AREQUIPA - PERÚ

2020

ÍNDICE GENERAL

1	Introducción	2
2	Métodos con alineación y sin alineación	2
3	MÉTODO	2
3.1	Objetivos	2
3.2	Algoritmo CASTOR-KRFE	3
3.3	Conjunto de datos	4
4	Implementación	5
4.1	Pasos principales	5
4.1.1	Composición y preprocesamiento de vectores de características	5
4.1.2	Identificación de conjuntos de características y su evaluación	6
4.1.3	Extracción de los k-mers	6
4.2	Código Completo	6
5	Resultados	18
6	REPOSITORIO	19

1 INTRODUCCIÓN

La clasificación de secuencias genómicas es una práctica fundamental en diferentes campos de la investigación en microbiología. Consiste en asignar una secuencia dada a su grupo relacionado de secuencias conocidas que comparten características y rasgos similares.

CASTOR-KRFE es un método sin alineación que extrae características discriminantes de longitud óptima a partir de secuencias genómicas. Construye un modelo de predicción con una evaluación de éste. El modelo se puede utilizar para realizar la predicción de nuevas secuencias.

2 MÉTODOS CON ALINEACIÓN Y SIN ALINEACIÓN

Se implementaron varios métodos en herramientas para clasificar secuencias virales. Los métodos iniciales se basan en la alineación de secuencias que incluyen (1) herramientas de propósito general para la búsqueda de similitudes, como BLAST (Altschul et al., 1997) y USEARCH (Edgar, 2010) y (2) herramientas específicas de organismos como REGA (De Oliveira et al. al., 2005) y SCUEAL (Pond et al., 2009) para la clasificación del virus de inmunodeficiencia humana 1 (VIH-1). Sin embargo, el uso de métodos basados en alineación podría enfrentar ciertas limitaciones. De hecho, consumen mucho tiempo y memoria, y su carga computacional para grandes bases de datos presenta un cuello de botella real (Bonham-Carter et al., 2013 ; Zielezinski et al., 2017). Los métodos basados en alineación no son adecuados para genomas, como los de virus, que están expuestos a grandes variaciones genéticas como recombinación, mezcla y transferencia de genes horizontal (Duffy et al., 2008 ; Zielezinski et al., 2017). Además, realizar una alineación requiere a menudo el ajuste de varios parámetros (matrices de sustitución, penalizaciones por huecos, valores de umbral para parámetros estadísticos, etc.) que dependen del conocimiento a priori de la evolución de las secuencias que se comparan.

Por lo tanto, para superar estos problemas, los métodos sin alineación surgieron como una alternativa en la comparación y clasificación de secuencias. Transforman secuencias biológicas en vectores de características (Vinga, 2014) para calcular una distancia y construir un modelo filogenético o de aprendizaje automático (Xing et al., 2010). El manejo de secuencias en un espacio vectorial evita correspondencias de residuos, lo que aumenta la velocidad de cálculo y proporciona robustez para la predicción de la variación genética (Zielezinski et al., 2017). Para obtener una clasificación precisa utilizando enfoques sin alineación, es obligatorio identificar las características relevantes.

Hoy en día, los k-mers (subsecuencias de nucleótidos de longitud k) constituyen una alternativa o complementariedad para vectorizar secuencias de nucleótidos (Bonham-Carter et al., 2013). El uso de este tipo de atributo ha demostrado su eficacia en la clasificación de secuencias a través de árboles filogenéticos (Blaisdell, 1986 ; Sims et al., 2009 ; Kolekar et al., 2012), enfoques basados en la distancia (Otu y Sayood, 2003 ; Liu et al., 2011 ; Nalbantoglu et al., 2011), o métodos basados en estadísticas y composiciones de nucleótidos (Ulitsky et al., 2006 ; Reinert et al., 2009 ; Chan et al., 2012).

3 MÉTODO

3.1 OBJETIVOS

En este trabajo se presenta a CASTOR-KRFE (extracción de k-mers por eliminación de características recursivas) para construir un conjunto de características relevantes para la clasificación de secuencias genómicas. Con este método, esperamos lograr tres objetivos principales: primero, extraer un conjunto mínimo de discriminaciones de subsecuencias maximizando los rendimientos de clasificación viral; segundo, identificar la longitud óptima de las subsecuencias maximizando la precisión de la clasificación; finalmente, haga que estas características sean fácilmente interpretables a escala humana. De hecho, varias subsecuencias extraídas podrían corresponder a información biológica significativa.

3.2 ALGORITMO CASTOR-KRFE

El algoritmo de aprendizaje CASTOR-KRFE toma como entrada (1)S , un conjunto de n secuencias conocidas, tal que, donde s_i es una cadena de caracteres de un alfabeto finito, y (2)y, un vector de etiquetas correspondientes a las clases de cada secuencia. k-mers se definen como las subsecuencias superpuestas de una secuencia s_i con longitudes de k. Ahora deja y ser, respectivamente, las longitudes mínima y máxima de k-mers y K el conjunto de longitudes a explorar de manera que el número de k-mers posibles está teóricamente limitado. El objetivo de CASTOR-KRFE es identificar el conjunto mínimo de k-mers con la longitud óptima de k maximizando la clasificación de cada secuencia de S dentro de las clases y según la puntuación de la medida F. El algoritmo CASTOR-KRFE se puede dividir en tres componentes principales. El primer componente es la composición y preprocessamiento de vectores de características. El segundo es la identificación de conjuntos de características y su evaluación. El tercero corresponde a la extracción de los k-mers que representan los conjuntos de características óptimos para construir un modelo de predicción. En el algoritmo 1 , para cada uno de los conjuntos k-mers presentes en S y se calcula una matriz de ocurrencias X. X contiene, para cada secuencia, el número de apariciones de k-mers. A escala min max entre 0 y 1 se aplica entonces a X . Como se menciona en Hsu et al. (2003), permite evitar el dominio de características con rangos numéricos mayores sobre aquellas con rangos numéricos menores y disminuye las dificultades numéricas durante el cálculo. Además, la escala min-max ha demostrado su capacidad para lograr un mejor rendimiento que otros métodos de normalización (Al Shalabi et al., 2006).

Después de este paso de preprocessamiento, el algoritmo comenzará analizando si el número actual de características (corresponde al número de columnas en la matriz X) es mayor que el número máximo de características dado en el parámetro (). Si este es el caso, se llama a una selección preliminar con eliminación de características recursivas basada en la máquina de vectores de soporte (SVM-RFE) (Guyon et al., 2002). La clasificación se basa en los pesos asignados al clasificador. Para cada iteración, se elimina un número determinado de funciones hasta alcanzar el número de funciones objetivo. Para este SVM-RFE preliminar, se elige un paso de eliminación predeterminado del 10% para eliminar rápidamente la mayoría de las características innecesarias y retener solo características. Una vez que se completa este paso preliminar, para cada f que van desde a, SVM-RFE se aplicará para reducir el número de características a f mediante un paso de eliminación de 1. Cada conjunto de características seleccionado se evaluará mediante una validación cruzada de pliegues x estratificada, métricas de rendimiento (medida F ponderada) y una clasificador (SVM), donde x se puede dar como entrada.

Algorithm 1: CASTOR-KRFE: Features extractor

```

Input: S: labeled nucleotide sequences,
        $k_{\min}$ : minimum length of k to identify,
        $k_{\max}$ : maximum length of k to identify,
        $f_{\min}$ : minimum number of features,
        $f_{\max}$ : maximum number of features,
Output:  $f_{\text{list}}$  list of potential feature set,
         $s_{\text{list}}$ : scores assigned to  $f_{\text{list}}$ 
1 Begin
2    $f_{\text{list}} \leftarrow \emptyset$ 
3    $s_{\text{list}} \leftarrow \emptyset$ 
4   foreach  $k \in [k_{\min} \dots k_{\max}]$  do
5      $D \leftarrow k\text{-mers} \in S$ 
6     foreach  $s_i \in S$  do
7        $X \leftarrow$  occurrences of each  $k$ -mers  $\in D$ 
8     end
9      $X \leftarrow \text{MinMaxScaler}(X, 0, 1)$ 
10    if  $\text{number\_of\_features}(X) > f_{\max}$ , then
11       $X \leftarrow \text{RFE}(X, f_{\max})$            #Apply RFE to obtain  $f_{\max}$  features
12      foreach  $f \in [f_{\min} \dots f_{\max}]$  do
13         $X \leftarrow \text{RFE}(X, f)$ 
14         $f_{\text{list}} \leftarrow \text{features}(X)$ 
15         $s_{\text{list}} \leftarrow \text{Cross-Validation}(X)$ 
16      end
17    end
18 End
```

Figure 3.1: LEBATTEUX ET AL 2019

Algorithm 2: CASTOR-KRFE: Optimal set identifier and prediction model builder

Input: f_{list} : list of extracted feature sets,
 s_{list} : scores assigned to f_{list} ,
 T : percentage performance threshold
 Algorithm: supervised learning algorithm

Output: k_{length} : optimal length of k ,
 f_{set} : optimal set of k -mers,
 model: predictive model based on f_{set}

```
1 Begin
2   bestscore ← max( $s_{list}$ )
3    $f_{set} \leftarrow$  feature set  $\in f_{list}$  associated with  $bestscore$ 
4    $n \leftarrow len(f_{set})$ 
5    $k_{length} \leftarrow len(features\ of\ f_{set})$ 
6   foreach  $f, s \in f_{list}, s_{list}$  do
7     if  $s > bestscore * T$  and  $len(f) < n$  then
8        $f_{set} \leftarrow f$ 
9        $n \leftarrow len(f)$ 
10       $k_{length} \leftarrow len(features\ of\ f)$ 
11    end
12  end
13  model ← (Algorithm,  $f_{set}$ )
14 End
```

Figure 3.2: LEBATTEUX ET AL 2019

Los conjuntos de características y sus puntuaciones asociadas se incluirán en los extractores de características y en la evaluación del clasificador como se describe en el algoritmo 2. El algoritmo utiliza una SVM basada en un kernel lineal. Este núcleo requiere unos pocos hiperparámetros que implican una baja complejidad del modelo de selección. En la última parte del algoritmo 2 , analizamos las listas de puntuación de rendimiento para extraer el conjunto de características óptimas y la longitud k óptima de las subsecuencias. La característica óptima seleccionada debe satisfacer dos condiciones. Primero, tiene una puntuación de medida F más alta que la mejor puntuación multiplicada por el umbral de rendimiento T(porcentaje de rendimiento mantenido en relación con la mejor puntuación). En segundo lugar, tiene la menor cantidad de funciones. Finalmente, se entrena un algoritmo de aprendizaje supervisado con el conjunto de características óptimas para construir un modelo predictivo.

3.3 CONJUNTO DE DATOS

Se aplica el algoritmo a un conjunto heterogéneo de datos genómicos que cubren los siete principales grupos de virus. De una amplia gama de conjuntos de datos, tomamos muestras de virus reemergentes que afectan a las poblaciones mundiales actuales, como el virus del Ébola, el VIH, el virus de la hepatitis C (VHC) y el virus del dengue. La identificación precisa y rápida de este tipo de virus puede tener importantes impactos para su estudio y la preservación de la vida y la salud humanas. De hecho, la enfermedad por el virus del Ébola se asocia con una tasa de letalidad del 30% al 90%, según la especie del virus (Baize et al., 2014). El virus del dengue también se ha incluido en nuestro conjunto de datos. La fiebre del dengue es una enfermedad emergente en todas las regiones tropicales y subtropicales de Asia, África, América y el Pacífico (Vaughn et al., 2000). Cada año, se estiman al menos 100 millones de infecciones por dengue (Halstead, 1988). La infección por dengue se asocia con fiebre hemorrágica y síndromes de choque, que es una de las principales causas de morbilidad y mortalidad pediátricas en Asia tropical (sangkawibha et al., 1984).

Algunos virus reemergentes, al mismo tiempo, son complejos y con una gran variedad de subtipos y altas tasas de recombinación. De hecho, el VHC incluye muchos genotipos que difieren entre sí en aproximadamente un 30% a nivel de nucleótidos. Además, cada genotipo está compuesto por múltiples subtipos divergentes de 20% en su composición de nucleótidos (Simmonds et al., 2005). La clasificación del VIH también representa un gran interés. De hecho, su grupo predominante M se divide actualmente en 13 subtipos. La variación genética dentro de un subtipo puede oscilar entre el 15% y el 20%, mientras que la variación entre subtipos suele ser del 25% al 35% (Taylor et al., 2008). Además, el VIH tiene altas tasas de mutación y recombinación. Son dos características esenciales del ciclo de replicación que permiten la propagación continua del VIH en todo el mundo, lo que implica una pandemia de complejidad genética y geográfica sin precedentes (Taylor et

al., 2008).

4 IMPLEMENTACIÓN

Se establecen los parametros para los calculos.

```
1
2
3 # VARIABLES
4
5 # Umbral (porcentaje de perdida de rendimiento en términos de medida F para reducir el
   ↪ número de atributos)
6 T = 0.999
7 # Longitud mínima de k-mer(s)
8 k_min = 1
9 # Longitud máxima de k-mer(s)
10 k_max = 5
11 # Número mínimo de características para identificar
12 features_min = 1
13 # Número máximo de características para identificar
14 features_max = 100
15 # Ruta de archivo fasta de entrenamiento
16 training_fasta = "Input/HPV_Alpha_species_sample_1.fa"#"Input/HIVGRPCG/data.fasta"
17 # Ruta de archivo fasta de entrenamiento
18 training_csv = "Input/HPV_Alpha_species_sample_1.csv"#"Input/HIVGRPCG/target.csv"
19 # Ruta de archivo fasta de pruebas
20 testing_fasta = "Input/HPV_Alpha_species_sample_1.fa"#"Input/HIVGRPCG/data.fasta"
21 # Ruta de archivo fasta de pruebas
22 testing_csv = "Input/HPV_Alpha_species_sample_1.csv"#"Input/HIVGRPCG/target.csv"
```

4.1 PASOS PRINCIPALES

4.1.1 COMPOSICIÓN Y PREPROCESAMIENTO DE VECTORES DE CARACTERÍSTICAS

En esta sección se realiza la comprobación de la existencia y accesibilidad de archivos de entrenamiento(archivo fasta) y de las etiquetas(archivos csv) como también de los archivos de pruebas. Se da el inicio de la extracción de características para k-mers. Ademas de la vectorización de secuencias genómicas virales conocidas basadas en k-mers para constituir las características potenciales. Y la generación del dato de entrenamiento.

Listing 1: Comprobación de Datos

```
1
2
3 # CARGAR DATOS DE ENTRENAMIENTO
4 print("\nLoading of the training dataset...")
5 if Data.checkTrainFile(training_fasta, training_csv) == True: training_data = Data.generateTrainData(
   ↪ training_fasta, training_csv)
6
7
8 # EXTRACCION DE CARACTERISTICAS
9 print("\nStart feature extraction...")
10 extracted_k_mers, identified_k_length = Extraction.extractKmers(T, training_data, k_min, k_max,
   ↪ features_min, features_max)
```

4.1.2 IDENTIFICACIÓN DE CONJUNTOS DE CARACTERÍSTICAS Y SU EVALUACIÓN

En esta sección se realiza la evaluación del modelo, mediante prueba cruzada y se cargan los datos de prueba para medir la precisión del clasificador. Forma eficiente de extracción y evaluación de patrones que maximizan el rendimiento de clasificación

```
1
2
3 # EVALUACION DEL MODELO
4 print("\nEvaluation of the prediction model...")
5 Evaluation.cross_validation(training_data, extracted_k_mers, identified_k_length, training_data)
6
7
8 # CARGAR DATOS DE PRUEBA
9 print("\nLoading of the testing dataset...")
10 if Data.checkTestFile(testing_fasta, testing_csv) == True: testing_data = Data.generateTestData(
    ↪ testing_fasta, testing_csv)
```

4.1.3 EXTRACCIÓN DE LOS K-MERS

Predicción del conjunto mínimo de características que se ajustan a un criterio dado (umbral de métrica de rendimiento y número máximo de características). Se obtiene el valor de longitud óptima para K.

```
1
2
3 # PREDICCION
4 if len(testing_data[0]) == 2:
5     print("\nPrediction without evaluation...")
6     Evaluation.prediction(training_data, testing_data, extracted_k_mers, identified_k_length)
7 else:
8     print("\nPrediction with evaluation...")
9     Evaluation.predictionEvaluation(training_data, testing_data, extracted_k_mers, identified_k_length
    ↪ )
10
11
12 print("\nEnd of the program ")
```

4.2 CODIGO COMPLETO

```
1
2 # IMPORTS
3 import Data
4 import Extraction
5 import Evaluation
6
7 # INFORMACION
8 print("*****")
9 print("*** CASTOR-KRFE ***")
10 print("*****\n")
11
12 print("Alignment-free method to extract discriminant genomic subsequences within pathogen
    ↪ sequences.\n")
13
14 # VARIABLES
```

```

15
16 # Umbral (porcentaje de perdida de rendimiento en términos de medida F para reducir el
17 # → número de atributos)
17 T = 0.999
18 # Longitud mínima de k-mer(s)
19 k_min = 1
20 # Longitud máxima de k-mer(s)
21 k_max = 5
22 # Número mínimo de características para identificar
23 features_min = 1
24 # Número máximo de características para identificar
25 features_max = 100
26 # Ruta de archivo fasta de entrenamiento
27 training_fasta = "Input/HPV_Alpha_species_sample_1.fa"#"Input/HIVGRPCG/data.fasta"
28 # Ruta de archivo fasta de entrenamiento
29 training_csv = "Input/HPV_Alpha_species_sample_1.csv"#"Input/HIVGRPCG/target.csv"
30 # Ruta de archivo fasta de pruebas
31 testing_fasta = "Input/HPV_Alpha_species_sample_1.fa"#"Input/HIVGRPCG/data.fasta"
32 # Ruta de archivo fasta de pruebas
33 testing_csv = "Input/HPV_Alpha_species_sample_1.csv"#"Input/HIVGRPCG/target.csv"
34
35
36 # CARGAR DATOS DE ENTRENAMIENTO
37 print("\nLoading of the training dataset...")
38 if Data.checkTrainFile(training_fasta, training_csv) == True: training_data = Data.generateTrainData(
    → training_fasta, training_csv)
39
40
41 # EXTRACCION DE CARACTERISTICAS
42 print("\nStart feature extraction...")
43 extracted_k_mers, identified_k_length = Extraction.extractKmers(T, training_data, k_min, k_max,
    → features_min, features_max)
44
45
46 # EVALUACION DEL MODELO
47 print("\nEvaluation of the prediction model...")
48 Evaluation.cross_validation(training_data, extracted_k_mers, identified_k_length, training_data)
49
50
51 # CARGAR DATOS DE PRUEBA
52 print("\nLoading of the testing dataset...")
53 if Data.checkTestFile(testing_fasta, testing_csv) == True: testing_data = Data.generateTestData(
    → testing_fasta, testing_csv)
54
55
56 # PREDICCION
57 if len(testing_data[0]) == 2:
58     print("\nPrediction without evaluation...")
59     Evaluation.prediction(training_data, testing_data, extracted_k_mers, identified_k_length)
60 else:
61     print("\nPrediction with evaluation...")
62     Evaluation.predictionEvaluation(training_data, testing_data, extracted_k_mers, identified_k_length
        → )
63

```

```

64
65 print("\nEnd of the program ")

```

```

1 # Imports
2 import os
3 import sys
4 import csv
5 from Bio import SeqIO
6
7 # Funcin de comprobacin de la existencia y accesibilidad de archivos de formacin.
8 def checkTrainFile(training_fasta, training_csv):
9     # Ver archivo fasta
10    if os.path.isfile(training_fasta) and os.access(training_fasta, os.R_OK):
11        print(training_fasta, "file exists and is readable")
12    # Comprobar archivo csv
13    if os.path.isfile(training_csv) and os.access(training_csv, os.R_OK):
14        print(training_csv, "file exists and is readable")
15    # Devuelve verdadero si todo es correcto
16    return True
17    # Salir y mostrar un mensaje en caso de error
18    else: sys.exit("Training csv file is missing or not readable")
19 else: sys.exit("Training fasta file is missing or not readable")
20
21 # Funcin que comprueba la existencia y accesibilidad de los archivos de prueba.
22 def checkTestFile(testing_fasta, testing_csv):
23     # Ver archivo fasta
24     if os.path.isfile(testing_fasta) and os.access(testing_fasta, os.R_OK):
25         print(testing_fasta, "file exists and is readable")
26     # Comprobar archivo csv
27     if os.path.isfile(testing_csv) and os.access(testing_csv, os.R_OK):
28         # Devuelve True si todo es correcto (prediccin con evaluacin)
29         print(testing_csv, "file exists and is readable")
30         return True
31     else:
32         # Devuelve True si solo el archivo fasta es correcto (prediccin sin evaluacin)
33         print("Testing csv file is missing or not readable")
34         return True
35 else: sys.exit("Testing fasta file is missing or not readable")
36
37 # Funcin que genera la tabla de datos
38 def generateTrainData(fasta_file, csv_file):
39     # Variable data
40     data = []
41
42     # Abrir el archivo de la clase
43     with open(csv_file) as f: reader = dict(csv.reader(f))
44
45     #Abrir el archivo de secuencias
46     for record in SeqIO.parse(fasta_file, "fasta"):
47         # Generar tabla [Id, Sequences, Class]
48         if record.id in reader: data.append([record.id, record.seq.upper(), reader[record.id]])
49
50     # Return data
51     return data

```

```

52
53 # Funcin que genera la tabla de datos
54 def generateTestData(fasta_file, csv_file):
55     # Variable data
56     data = []
57
58     # Llamar a la funcin clsica
59     if csv_file: data = generateTrainData(fasta_file, csv_file)
60     else:
61         # Abra el archivo de secuencias y genere la tabla [Id, Sequences]
62         for record in SeqIO.parse(fasta_file, "fasta"): data.append([record.id, record.seq.upper()])
63
64     # Return data
65     return data

```

```

1 # Imports
2 import joblib
3 import Matrices
4 import Preprocessing
5 from sklearn import svm
6 from sklearn.metrics import confusion_matrix
7 from sklearn.metrics import classification_report
8 from sklearn.model_selection import StratifiedKFold
9 from sklearn.model_selection import cross_val_predict
10
11 # Funcin de evaluacin del modelo
12 def cross_validation(training_data, extracted_k_mers, identified_k_length, data):
13     # Generar matrices
14     X, y = Matrices.generateMatrice(training_data, extracted_k_mers, identified_k_length)
15     X = Preprocessing.minMaxScaling(X)
16
17     # Realice la evaluacin con CV + Clasificador + Mtricas
18     classifier = svm.SVC(kernel = 'linear', C = 1)
19     stratifiedKFold = StratifiedKFold(n_splits = 5, shuffle = False, random_state = None)
20     y_pred = cross_val_predict(classifier, X, y, cv = stratifiedKFold, n_jobs = 4)
21
22     # Imprimir resultados de la evaluacin del modelo
23     classificationReport = classification_report(y, y_pred, digits = 3)
24     confusionMatrix = confusion_matrix(y, y_pred)
25     print("\nClassification report of model evaluation\n", classificationReport)
26     print("Confusion matrix \n", confusionMatrix)
27
28     # Guardar Matrice
29     f = open("Output/Matrice.csv", "w")
30     f.write("Id,")
31     for i in extracted_k_mers: f.write(str(i) + ",")
32     f.write("Class\n")
33
34     for i, x in enumerate(X):
35         f.write(str(data[i][0]) + ",")
36         for j in x: f.write(str(j) + ",")
37         f.write(str(y[i]) + "\n")
38     f.close()
39

```

```

40 # Guardar model
41 classifier.fit(X, y)
42 joblib.dump(classifier, 'Output/model.pkl')
43
44 # Guardar los resultados de la evaluacion del modelo
45 f = open("Output/Model_Evaluation.txt", "w")
46 f.write("Classification report of model evaluation\n" + classificationReport);
47 f.write("\nConfusion matrix \n" + str(confusionMatrix));
48 f.close()
49
50
51
52 # Funcin de prediccin sin evaluacin
53 def prediction(training_data, testing_data, extracted_k_mers, identified_k_length):
54     # Generar matrices
55     X_test, y_test = Matrices.generateMatrice(testing_data, extracted_k_mers, identified_k_length)
56     X_test = Preprocessing.minMaxScaling(X_test)
57
58     # Cargar model
59     classifier = joblib.load('Output/model.pkl')
60
61     # Realizar prediccin
62     y_pred = classifier.predict(X_test)
63
64     # Guardar prediccin
65     f = open("Output/Prediction.csv", "w")
66     f.write("id,y_pred\n");
67     for i, y in enumerate(y_pred): f.write(testing_data[i][0] + "," + y + "\n");
68     f.close()
69
70 # Funcin de prediccin con evaluacin
71 def predictionEvaluation(training_data, testing_data, extracted_k_mers, identified_k_length):
72     # Generar matrices
73     X_test, y_test = Matrices.generateMatrice(testing_data, extracted_k_mers, identified_k_length)
74     X_test = Preprocessing.minMaxScaling(X_test)
75
76     # Cargar model
77     classifier = joblib.load('Output/model.pkl')
78
79     # Realizar prediction
80     y_pred = classifier.predict(X_test)
81
82     # Imprimir resultados
83     classificationReport = classification_report(y_test, y_pred, digits = 3)
84     confusionMatrix = confusion_matrix(y_test, y_pred)
85     print("\nClassification report of prediction evaluation\n", classificationReport)
86     print("Confusion matrix \n", confusionMatrix)
87
88     # Guardar prediccin
89     f = open("Output/Prediction_Evaluation.csv", "w")
90     f.write("id,y_pred,y_true\n");
91     for i, y in enumerate(y_pred): f.write(testing_data[i][0] + "," + y + "," + y_test[i] + "\n");
92     f.close()
93

```

```

94 # Guardar los resultados de la evaluacin de la prediccin
95 f = open("Output/Prediction_Evaluation.txt", "w")
96 f.write("Classification report of prediction evaluation\n" + classificationReport);
97 f.write("\nConfusion matrix \n" + str(confusionMatrix));
98 f.close()

```

```

1 # Imports
2 import numpy
3 import K_mers
4 import Matrices
5 import Preprocessing
6 from sklearn import svm
7 import matplotlib.pyplot as plt
8 from sklearn.metrics import f1_score
9 from sklearn.feature_selection import RFE
10 from sklearn.model_selection import StratifiedKFold
11 from sklearn.model_selection import cross_val_predict
12
13 # Funcin bsica de extraccin de caractersticas
14 def extractKmers(T, training_data, k_min, k_max, features_min, features_max):
15     # Contiene listas de puntuaciones de diferentes longitudes de k
16     scores_list = []
17     # Contiene una lista de k-mer para cada iteracin de rfe
18     supports_list = []
19     # Lista de diferentes longitudes de k-mer
20     k_mers_range = range(k_min, k_max + 1)
21     # Clasificador svm
22     classifier = svm.SVC(kernel = 'linear', C = 1)
23
24     # Realizar el anlisis para los diferentes tamaos de k
25     for k in k_mers_range:
26         # Iniciar extraccin de caractersticas basada en k-mers de longitud k
27         print("\nBeginning of the " + str(k) + " _mer(s) analysis")
28
29         # Generarte lista de k-mer
30         print("Generate K-mers...")
31         k_mers = K_mers.generate_K_mers(training_data, k)
32
33         # Generar atributos de matrices y clases de matrices
34         print("Generate matrices...")
35         X, y = Matrices.generateMatrice(training_data, k_mers, k)
36         y = numpy.asarray(y)
37
38         # Aplicar MinMaxScaler (0, 1)
39         X = Preprocessing.minMaxScaling(X)
40
41         # Si hay ms de features_max, aplique RFE (elimine el 10% de las caractersticas para
42             # eliminar en cada iteracin)
43         if len(X[0]) > features_max:
44             print("Preliminary recursive feature elimination...")
45             rfe = RFE(estimator = classifier, n_features_to_select = features_max, step = 0.1)
46             X = numpy.matrix(X)
47             X = rfe.fit_transform(X, y)

```

```

48     # Actualizar lista de k_mers
49     for i, value in enumerate(rfe.support_):
50         if value == False: k_mers[i] = None
51     k_mers = list(filter(lambda a: a != None, k_mers))
52
53     # Eliminacion de caracteristicas recursivas
54     from RFE import RFE
55     print("Recursive feature elimination...")
56     rfe = RFE(estimator = classifier, n_features_to_select = 1, step = 1)
57     rfe.fit(X,y)
58
59     # Puntajes y apoyos de la iteracion actual
60     scores = []
61     supports = []
62
63     # Evaluacion
64     for i, supports_rfe in enumerate(rfe.supports):
65         # Variables
66         temp_index = []
67         temp_k_mers = []
68
69         # Imprimir porcentaje de avance
70         print("\rFeature subset evaluation : ", round((i + 1) / len(rfe.supports) * 100, 0), "%",
71             end = ',')
72
73         # Selecciona k-mers con soporte igual True
74         for j, support in enumerate(supports_rfe):
75             if rfe.supports[i][j] == True: temp_index.append(j)
76
77         # Reemplazar el soporte por los k-mers
78         for t in temp_index: temp_k_mers.append(k_mers[t])
79         rfe.supports[i] = temp_k_mers
80
81         # Metodo de evaluacion
82         stratifiedKFold = StratifiedKFold(n_splits = 5, shuffle = False, random_state = None)
83         y_pred = cross_val_predict(classifier, X[:,temp_index], y, cv = stratifiedKFold, n_jobs = 4)
84         score = f1_score(y, y_pred, average = 'weighted')
85
86         # Guardar la puntuacion y las caracteristicas de esta iteracion
87         scores.append(score)
88         supports.append(rfe.supports[i])
89
90         # Guarde la lista de puntuaciones y subconjuntos de funciones para esta longitud de
91             # k-mers
92         scores_list.append(scores)
93         supports_list.append(supports)
94
95     # Cambia el orden de las listas para el grafico.
96     for i, e in enumerate(scores_list):
97         scores_list[i].reverse()
98         supports_list[i].reverse()
99
# Identificar solucion
print("\n\nIdentify optimal solution...")

```

```

100 # Mejor puntuacin de las evaluaciones
101 best_score = 0
102 # Puntuacin ptima en relacin con el threshold
103 optimal_score = 0
104 # Mejor lista de k-mer
105 extracted_k_mers = []
106 # Mejor longitud de k
107 identified_k_length = 0
108
109 # Identificar la mejor solucin
110 for i, s in enumerate(scores_list):
111     if max(s) > best_score:
112         best_score = max(s)
113         index = s.index(max(s))
114         identified_k_length = k_mers_range[i]
115         extracted_k_mers = supports_list[i][index]
116     elif max(s) == best_score:
117         if s.index(max(s)) < index:
118             best_score = max(s)
119             index = s.index(max(s))
120             identified_k_length = k_mers_range[i]
121             extracted_k_mers = supports_list[i][index]
122     else: pass
123
124 # Identificar la solucin ptima
125 for i, l in enumerate(scores_list):
126     for j, s in enumerate(l):
127         if s >= best_score * T and j <= index:
128             optimal_score = s
129             index = j
130             identified_k_length = k_mers_range[i]
131             extracted_k_mers = supports_list[i][index]
132 if optimal_score == 0: optimal_score = best_score
133
134
135 # Guardar los resultados del grfico
136 fig = plt.figure(figsize = (12, 10) )
137 for i, s in enumerate(scores_list):
138     label = str(k_mers_range[i]) + "-mers"
139     plt.plot(range(len(s)), s, label = label)
140 plt.ylabel("F-measure")
141 plt.xlabel("Number of features")
142 plt.axvline(index + 1, linestyle = ':', color = 'r')
143 title = "F-measure : " + str(round(optimal_score, 3)) + " K-mer size : " + str(
    ↪ identified_k_length) + " Number of features : " + str(index + 1)
144 plt.title(title)
145 plt.legend()
146 fname = str("Output/Analysis.png")
147 plt.savefig(fname)
148
149 # Guardar k-mers extrados
150 f = open("Output/Kmers.txt", "w")
151 for i in extracted_k_mers: f.write(str(i) + "\n");
152 f.close()

```

```

153
154     # Devuelve k-mers identificados y su longitud
155     return extracted_k_mers, identified_k_length

```

```

1 import re #se importa la libreria para trabajar con expresiones regulares
2
3 # Funcin que genera los k_mers pertenecientes a las secuencias
4 def generate_K_mers(data, k):
5     # List of k-mer
6     K_mers = []
7     dict = {}
8
9     # Inicializacin del diccionario
10    for d in data:
11        for i in range(0, len(d[1]) - k + 1, 1): dict[d[1][i:i + k]] = 0;
12
13    # Eliminar patrones no utilizados
14    for key in dict.keys():
15        if bool(re.match('^[ACGT]+$', str(key))) == True: K_mers.append(str(key))
16
17    # Retorna los kmers
18    return K_mers

```

```

1 # Funcin de generacin de matrices de clases y caractersticas
2 def generateMatrice(data, K_mer, k):
3     # Variables
4     X = []
5     y = []
6
7     # Generar diccionario K-mer
8     X_dict = {}
9     for i, e in enumerate(K_mer): X_dict[e] = 0;
10    # Generar X (atributos de matriz)
11    for d in data:
12        x = []
13        x_dict = X_dict.copy()
14
15        # Contar ocurrencias de K-mer (con superposicin)
16        for i in range(0, len(d[1]) - k + 1, 1):
17            try: x_dict[d[1][i:i + k]] = x_dict[d[1][i:i + k]] + 1;
18            except: pass
19
20        # Obtener todas las ocurrencias del diccionario
21        for value in x_dict:
22            x.append(x_dict.get(value))
23        X.append(x)
24
25    # Genera y (clase Matrix) si existe un archivo csv
26    if len(data[0]) == 3:
27        for i in data: y.append(i[2])
28
29    # Retornar matrices X e y (atributos de matriz y clase de matriz)
30    return X, y

```

```

1 from sklearn.preprocessing import MinMaxScaler
2
3 # Funcion ode MinMaxScaler (0, 1)
4 def minMaxScaling(X):
5     minMaxScaler = MinMaxScaler(feature_range = (0, 1), copy = False)
6     X = minMaxScaler.fit_transform(X)
7     return X

```

```

1 ######
2 # Authors: Alexandre Gramfort <alexandre.gramfort@inria.fr> #
3 # Vincent Michel <vincent.michel@inria.fr> #
4 # Gilles Louppe <g.louppe@gmail.com> #
5 #
6 # Modified by Dylan Lebatteux <lebatteux.dylan@courrier.uqam.ca> #
7 #
8 # License: BSD 3 clause #
9 #####
10
11 # Imports
12 import numpy as np
13 from sklearn.base import clone
14 from sklearn.utils import safe_sqr
15 from sklearn.metrics import f1_score
16 from sklearn.base import BaseEstimator
17 from sklearn.base import is_classifier
18 from sklearn.metrics import check_scoring
19 from sklearn.base import MetaEstimatorMixin
20 from sklearn.model_selection import check_cv
21 from sklearn.utils.metaestimators import _safe_split
22 from sklearn.utils.validation import check_is_fitted
23 from sklearn.model_selection._validation import _score
24 from joblib import Parallel, delayed, effective_n_jobs
25 from sklearn.feature_selection.base import SelectorMixin
26 from sklearn.utils.metaestimators import if_delegate_has_method
27 from sklearn.utils.validation import _deprecate_positional_args
28
29 def _rfe_single_fit(rfe, estimator, X, y, train, test, scorer):
30     X_train, y_train = _safe_split(estimator, X, y, train)
31     X_test, y_test = _safe_split(estimator, X, y, test, train)
32     return rfe._fit(X_train, y_train, lambda estimator, features: _score(estimator, X_test[:, features],
33                         ↪ y_test, scorer)).scores_
34
35 class RFE(SelectorMixin, MetaEstimatorMixin, BaseEstimator):
36     @_deprecate_positional_args
37     def __init__(self, estimator, *, n_features_to_select=None, step=1, verbose=0):
38         self.supports = []
39         self.estimator = estimator
40         self.n_features_to_select = n_features_to_select
41         self.step = step
42         self.verbose = verbose
43
44     @property
45     def _estimator_type(self): return self.estimator._estimator_type

```

```

45
46     @property
47     def classes_(self): return self.estimator_.classes_
48
49     def fit(self, X, y): return self._fit(X, y)
50
51     def _fit(self, X, y, step_score=None):
52         tags = self._get_tags()
53         X, y = self._validate_data(X, y, accept_sparse="csc", ensure_min_features=2, force_all_finite
54             ↪ =not tags.get('allow_nan', True), multi_output=True)
55
56         # Inicializacin
57         n_features = X.shape[1]
58         if self.n_features_to_select is None: n_features_to_select = n_features // 2
59         else: n_features_to_select = self.n_features_to_select
60
61         if 0.0 < self.step < 1.0: step = int(max(1, self.step * n_features))
62         else: step = int(self.step)
63         if step <= 0: raise ValueError("Step must be >0")
64
65         support_ = np.ones(n_features, dtype=np.bool)
66         ranking_ = np.ones(n_features, dtype=np.int)
67
68         if step_score: self.scores_ = []
69
70         # Eliminacin
71         while np.sum(support_) > n_features_to_select:
72             # Features restantes
73             features = np.arange(n_features)[support_]
74
75             # Clasifique las caractersticas restantes
76             estimator = clone(self.estimator)
77             if self.verbose > 0: print("Fitting estimator with %d features." % np.sum(support_))
78
79             # Ajuste
80             estimator.fit(X[:, features], y)
81
82             # Obtener coefs
83             if hasattr(estimator, 'coef_'): coefs = estimator.coef_
84             else: coefs = getattr(estimator, 'feature_importances_', None)
85             if coefs is None: raise RuntimeError("The classifier does not expose coef_or
86                 ↪ feature_importances_attributes")
87
88             # Obtener rangos
89             if coefs.ndim > 1: ranks = np.argsort(np.abs(coefs).sum(axis=0))
90             else: ranks = np.argsort(np.abs(coefs))
91
92             # Para rangos de casos dispersos es matriz
93             ranks = np.ravel(ranks)
94
95             # Elimina las peores caractersticas
96             threshold = min(step, np.sum(support_) - n_features_to_select)
97
98             # Guardar soporte de caractersticas seleccionadas

```

```

97     self.supports.append(list(support_))

98
99     # Calcule el puntaje de paso en la iteración de selección anterior porque el 'estimador' debe usar características que aún no se han eliminado
100    if step_score: self.scores_.append(step_score(estimator, features))
101    support_[features[ranks][:threshold]] = False
102    ranking_[np.logical_not(support_)] += 1
103
104    # Establecer atributos finales
105    features = np.arange(n_features)[support_]
106    self.estimator_ = clone(self.estimator)
107    self.estimator_.fit(X[:, features], y)
108
109    # Calcule la puntuación por pasos cuando solo quedan n características para seleccionar características
110    if step_score: self.scores_.append(step_score(self.estimator_, features))
111    self.n_features_ = support_.sum()
112    self.support_ = support_
113    self.ranking_ = ranking_
114
115    # Guardar soporte de características seleccionadas
116    self.supports.append(list(support_))
117
118    return self
119
120 @if_delegate_has_method(delegate='estimator')
121 def predict(self, X):
122     check_is_fitted(self)
123     return self.estimator_.predict(self.transform(X))
124
125 @if_delegate_has_method(delegate='estimator')
126 def score(self, X, y):
127     check_is_fitted(self)
128     return self.estimator_.score(self.transform(X), y)
129
130 def _get_support_mask(self):
131     check_is_fitted(self)
132     return self.support_
133
134 @if_delegate_has_method(delegate='estimator')
135 def decision_function(self, X):
136     check_is_fitted(self)
137     return self.estimator_.decision_function(self.transform(X))
138
139 @if_delegate_has_method(delegate='estimator')
140 def predict_proba(self, X):
141     check_is_fitted(self)
142     return self.estimator_.predict_proba(self.transform(X))
143
144 @if_delegate_has_method(delegate='estimator')
145 def predict_log_proba(self, X):
146     check_is_fitted(self)
147     return self.estimator_.predict_log_proba(self.transform(X))
148

```

```

149     def _more_tags(self):
150         estimator_tags = self.estimator._get_tags()
151         return {'poor_score': True, 'allow_nan': estimator_tags.get('allow_nan', True), '→ requires_y': True,}

```

5 RESULTADOS

```

Archivo Editar Ver Buscar Terminal Ayuda
→ Input git:(master) X ls
HIVGRPCG PMSHIV01 PMSHIV01_2
→ Input git:(master) X

```

Figure 5.1: Dataset de entrada

```

python -W ignore Main.py
Archivo Editar Ver Buscar Terminal Ayuda
(test) → CASTOR_KRFE git:(master) X python -W ignore Main.py
*****
*** CASTOR-KRFE ***
*****  

Alignment-free method to extract discriminant genomic subsequences within pathogen sequences.  

Loading of the training dataset...
Input/HIVGRPCG/data.fasta file exists and is readable
Input/HIVGRPCG/target.csv file exists and is readable  

Start feature extraction...  

Beginning of the 1_mer(s) analysis
Generate K-mers...
Generate matrices...

```

Figure 5.2: Ejecutando el clasificador

```

Classification report of prediction evaluation
      precision    recall  f1-score   support

HIV-1.M       1.000     1.000     1.000      32
HIV-1.N       1.000     1.000     1.000      11
HIV-1.O       1.000     1.000     1.000      32
HIV-1.P       1.000     1.000     1.000       5

accuracy          1.000      80
macro avg       1.000     1.000     1.000      80
weighted avg    1.000     1.000     1.000      80

Confusion matrix
[[32  0  0  0]
 [ 0 11  0  0]
 [ 0  0 32  0]
 [ 0  0  0  5]]

End of the program
(test) → CASTOR_KRFE git:(master) X └─

```

Figure 5.3: Resultados de la clasificación

```

Archivo Editar Ver Buscar Terminal Ayuda
→ CASTOR_KRFE git:(master) X cd Output
→ Output git:(master) X ls
Analysis.png Matrice.csv      model.pkl      Prediction_Evaluation.csv
Kmers.txt     Model_Evaluation.txt Prediction.csv  Prediction_Evaluation.txt
→ Output git:(master) X └─

```

Figure 5.4: Salida de la compilación

6 REPOSITORIO

<https://github.com/noeldiazven/Castor>

REFERENCES

- [1] D. Lebatteux, A. M. Remita, and A. B. Diallo, “Toward an alignment-free method for feature extraction and accurate classification of viral sequences,” Journal of Computational Biology, vol. 26, no. 6, pp. 519–535, 2019.



UNIVERSIDAD NACIONAL DE SAN AGUSTIN

ESCUELA PROFESIONAL DE
CIENCIA DE LA COMPUTACIÓN

COMPUTACIÓN MOLECULAR
BIOLÓGICA

**Uso del análisis de textura de
imágenes para encontrar
similitudes en la secuencia de
ADN**

Alumnos:

Alberto Visa Flores

Sergio Arcos Ponce

Gustavo Leon Paredes

Alfred Guardia Zenteno

Docente:

Mg. Vicente Machaca

18 de agosto de 2020

Índice

1. Introducción	2
2. GLCM	2
3. Transformar una secuencia de ADN en un vector digital	3
4. Características basadas en la matriz de co-ocurrencia	3
4.1. Código	4
5. Resultados	8
6. Conclusiones	9
7. Repositorio	9

1. Introducción

El análisis de similitud de secuencia es la técnica básica para construir árboles filogenéticos, que analizan las funciones de los genes y predicen las estructuras de las proteínas. La alineación de secuencia es la más utilizada y método de análisis de similitud intuitivo. Muchas secuencias de alineación.

2. GLCM

Para calcular eficazmente las propiedades del ADN secuencias y para realizar análisis de similitud, proponemos un método basado en la teoría de la matriz de co-ocurrencia de niveles de gris (GLCM), que es un método estadístico bien conocido y de uso común método en el análisis de la textura de la imagen. Definir y especificar valores de características para cada secuencia es útil para encontrar secuencias similares en bases de datos, especialmente cuando el La base de datos es muy grande y una comparación de secuencia uno por uno es pérdida de tiempo. GLCM puede definir y calcular características relacionadas con cada secuencia; estas características también pueden indicar similitudes entre secuencias. Por lo tanto, las características definidas pueden ser los valores clave en cada secuencia. Al ingresar una secuencia y encontrar sus secuencias similares, todo lo que se requiere es calcular su valor de característica y secuencias de salida que tienen valores de características similares a los de la secuencia de entrada. Esto puede ahorrar mucho tiempo de búsqueda en la base de datos.

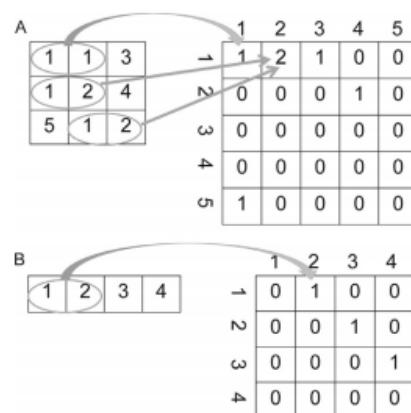


Figura 1: GLCM

3. Transformar una secuencia de ADN en un vector digital

Una secuencia de ADN está representada por una cadena que comprende A (adenina), C (citosina), G (guanina) y T (timina). En este documento, definir un método para transformar cada secuencia de ADN en un vector digital, que luego se puede utilizar para calcular la matriz de co-ocurrencia en la figura1. En el método propuesto, el procedimiento consiste en utilizar primero números enteros 1, 2, 3 y 4 para representar las cuatro bases A, C, G y T respectivamente. En segundo lugar, el número de cada carácter en la secuencia de ADN se suma al valor entero anterior.

4. Características basadas en la matriz de co-ocurrencia

Las características utilizadas fueron entropía, contraste, energía, correlación, y homogeneidad.

- **Entropia:**

$$\text{Entropy} = - \sum_{i=1}^L \sum_{j=1}^L p(i, j) \ln(p(i, j)),$$

- **Contraste:**

$$\text{Contrast} = \sum_{i=1}^L \sum_{j=1}^L (i - j)^2 p(i, j),$$

- **Energia:**

$$\text{Energy} = \sum_{i=1}^L \sum_{j=1}^L p(i, j)^2,$$

- **Correlacion:**

$$\text{Correlation} = \sum_{i=1}^L \sum_{j=1}^L \left(\frac{(i - \mu_i)(j - \mu_j)p(i, j)}{\sigma_i \sigma_j} \right),$$

- **Homogeneidad:**

$$Homogeneity = \sum_{i=1}^L \sum_{j=1}^L \left(\frac{p(i,j)}{1 + |i-j|} \right),$$

4.1. Código

```

1 from google.colab import drive
2 import re
3 import numpy as np
4 import re
5 def string_to_array(my_string):
6     my_string = my_string.lower()
7     my_string = re.sub('[^acgt]', 'z', my_string)
8     my_array = np.array(list(my_string))
9     return my_array
10 from sklearn.preprocessing import LabelEncoder
11 def ordinal_encoder(my_array):
12     label_encoder = LabelEncoder()
13     label_encoder.fit(np.array(['a', 'c', 'g', 't', 'z']))
14     integer_encoded = label_encoder.transform(my_array)
15     float_encoded = integer_encoded.astype(float)
16     float_encoded[float_encoded == 0] = 0 # A
17     float_encoded[float_encoded == 1] = 1 # C
18     float_encoded[float_encoded == 2] = 2 # G
19     float_encoded[float_encoded == 3] = 3 # T
20     float_encoded[float_encoded == 4] = 4 # anything else, z
21     return float_encoded
22 from sklearn.preprocessing import OneHotEncoder
23 def one_hot_encoder(my_array):
24     integer_encoded = label_encoder.transform(my_array)
25     onehot_encoder = OneHotEncoder(sparse=False, dtype=int, n_values=5)
26     integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
27     onehot_encoded = onehot_encoder.fit_transform(integer_encoded)
28     onehot_encoded = np.delete(onehot_encoded, -1, 1)
29     return onehot_encoded
30 def glcm(m_array):
31     a=npamax(m_array)
32     a=int(a)
33     template=np.zeros((a+1,a+1))
34     for i in range(m_array.shape[0]):
35         for j in range(m_array.shape[1]-1):
36             if m_array[i][j]!=-1. and m_array[i][j+1]!=-1.:
37                 template[int(m_array[i][j])][int(m_array[i][j+1])]+=1

```

```
38     return template
39 def glcm_vect(m_array):
40     template=np.zeros((4,4))
41     for i in range(m_array.shape[0]-1):
42         template[int(m_array[i])][int(m_array[i+1])]+=1
43     return template
44
45 def contrast(matriz_g):
46     resultado=0
47     for i in range(matriz_g.shape[0]):
48         for j in range(matriz_g.shape[1]):
49             resultado+=((i-j)**2)*matriz_g[i][j]
50     return resultado
51 def energy(matriz_g):
52     resultado=0
53     for i in range(matriz_g.shape[0]):
54         for j in range(matriz_g.shape[1]):
55             resultado+=matriz_g[i][j]
56     return resultado
57 def entropy(matriz_g):
58     a=np.log(matriz_g)
59     resultado=np.sum(a)
60     return resultado
61
62 from skimage.feature import greycomatrix, greycoprops
63 from multiprocessing import Pool
64 def glcm_props(patch):
65     lf = []
66     props = ['entropy', 'contrast', 'homogeneity', 'energy', 'correlation']
67
68     #para que vaya a la derecha np.pi/4
69     glcm = greycomatrix(patch, [1], [np.pi/4], 256, symmetric=True,
70                         normed=True)
71     for f in props:
72         lf.append(greycoprops(glcm, f)[0,0])
73
74     a = []
75     for f in props:
76         a.append(greycoprops(glcm, f)[0,0])
77
78 from Bio import SeqIO
79 data = []
```

```
80 for seq_record in SeqIO.parse('/content/drive/My Drive/molecular/  
example.fa', "fasta"):  
81     a=str(seq_record.seq)  
82     montar=string_to_array(a)  
83     montar=ordinal_encoder(montar)  
84     ggg=np.array(montar)  
85     m_glc=mglcm_vect(ggg)  
86     a=[]  
87     if int(ggg.shape[0] % 4) != 0:  
88         new_cont=(int(ggg.shape[0] // 4) + 1) * 4  
89         a=np.zeros(int(new_cont - ggg.shape[0]))  
90         ggg=np.append(ggg, a)  
91         f=ggg.shape[0]  
92         gope = np.array(ggg).reshape(int(f / 4), 4)  
93         a=glcm(gope)  
94         a=a.astype(np.uint8)  
95         gope=gope.astype(np.uint8)  
96         data.append(glcm_props(gope))  
97 for seq_record in SeqIO.parse('/content/drive/My Drive/molecular/  
Macaca_fascicularis_chromosome10.fa', "fasta"):  
98     a=str(seq_record.seq)  
99     montar=string_to_array(a)  
100    montar=ordinal_encoder(montar)  
101    ggg=np.array(montar)  
102    #m_glc=mglcm_vect(ggg)  
103    a=[]  
104    if int(ggg.shape[0] % 4) != 0:  
105        new_cont=(int(ggg.shape[0] // 4) + 1) * 4  
106        a=np.zeros(int(new_cont - ggg.shape[0]))  
107        ggg=np.append(ggg, a)  
108        f=ggg.shape[0]  
109        gope = np.array(ggg).reshape(int(f / 4), 4)  
110        a=glcm(gope)  
111        a=a.astype(np.uint8)  
112        gope=gope.astype(np.uint8)  
113        data.append(glcm_props(gope))  
114 print(data)  
115  
116 for seq_record in SeqIO.parse('/content/drive/My Drive/molecular/  
Macaca_mulatta_nonchromosomal.fa', "fasta"):  
117     a=str(seq_record.seq)  
118     montar=string_to_array(a)  
119     montar=ordinal_encoder(montar)  
120     ggg=np.array(montar)  
121     #m_glc=mglcm_vect(ggg)
```

```
122     a=[]
123     if int(ggg.shape[0] % 4) != 0:
124         new_cont = (int(ggg.shape[0] // 4) + 1) * 4
125         a = np.zeros(int(new_cont - ggg.shape[0]))
126         ggg = np.append(ggg, a)
127         f = ggg.shape[0]
128         gope = np.array(ggg).reshape(int(f / 4), 4)
129         a = glcm(gope)
130         a = a.astype(np.uint8)
131         gope = gope.astype(np.uint8)
132         data.append(glcm_props(gope))
133
134 for seq_record in SeqIO.parse('/content/drive/My Drive/molecular/
135 Macaca_mulatta_nonchromosomal.fa', "fasta"):
136     a = str(seq_record.seq)
137     montar = string_to_array(a)
138     montar = ordinal_encoder(montar)
139     ggg = np.array(montar)
140     #m_glcm = glcm_vect(ggg)
141     a = []
142     if int(ggg.shape[0] % 4) != 0:
143         new_cont = (int(ggg.shape[0] // 4) + 1) * 4
144         a = np.zeros(int(new_cont - ggg.shape[0]))
145         ggg = np.append(ggg, a)
146         f = ggg.shape[0]
147         gope = np.array(ggg).reshape(int(f / 4), 4)
148         a = glcm(gope)
149         a = a.astype(np.uint8)
150         gope = gope.astype(np.uint8)
151         data.append(glcm_props(gope))
```

5. Resultados

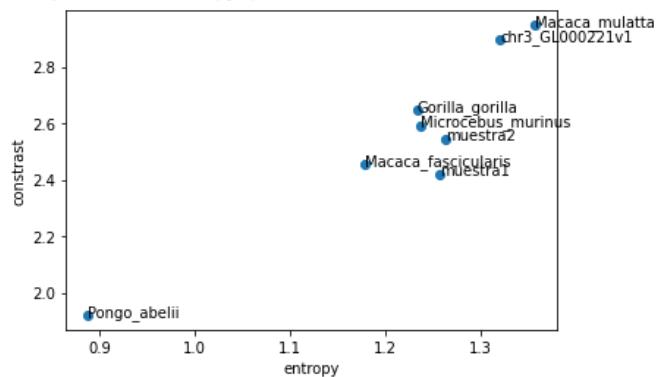


Figura 2: entropy vs contrast

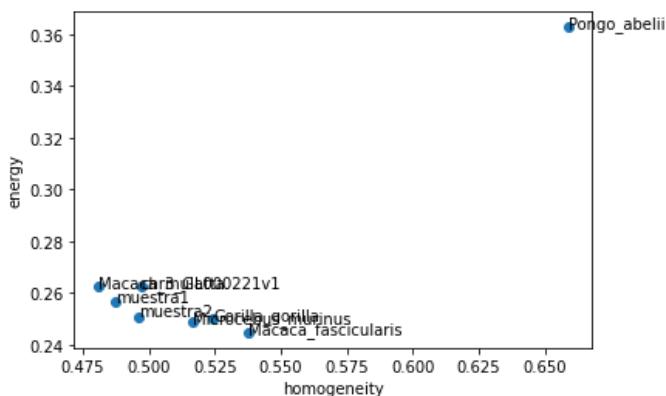


Figura 3: homogeneity vs entropy

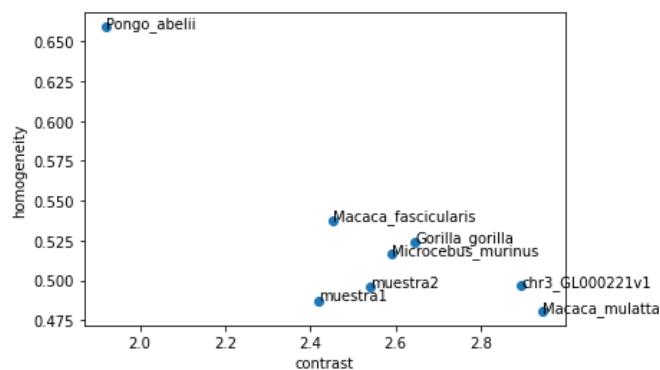


Figura 4: contrast vs homogeneity

6. Conclusiones

- Con este metodo podemos ver de manera grafica la similitud que hay entre una secuencia y otra.
- Se utiliza GLCM para sacar una matriz de texturas para cada secuencia,realizando las operaciones de entropia, contraste, energia, correlación, homogeneidad.

7. Repositorio

https://github.com/widcatd/molecular_glc

Referencias

- [1] Weiyang Chena, Bo Liao , Weiwei Li ,2018, Use of image texture analysis to find DNA sequence similarities,doi:10.1016/j.jtbi.2018.07.001

7.1 Evidencias

Rindieron la Tercera Evaluación Parcial 16 estudiantes de los 17 estudiantes matriculados, lo que representa un 100.0%.

La nota promedio de los estudiantes que rindieron la Tercera Evaluación Parcial es 17 puntos. Las notas y la evidencia de la Tercera Evaluación Parcial se encuentran en la siguiente tabla:

Tabla 7.1: Resultados Tercera Evaluación Parcial

Apellidos y Nombres	Grupo	EP 3	Evidencia
ARCOS/PONCE, SERGIO MANUEL	1	18	Enlace
BERMUDEZ/NAVARRO, WILLIAN BRAULIO	4	0	Enlace
CAYLLAHUE/CCORA, RENZO AUGUSTO	2	18	Enlace
CHAMBI APAZA, SYOMIRA INES	3	18	Enlace
CHAVEZ LOPEZ CAROLINA BONNIE	2	18	Enlace
CONDORI/MANSILLA, WILLIAM SILVERIO	2	18	Enlace
DEXTRE/AIQUIPA, MARKS CRISTOPHER	3	18	Enlace
DIAZ/VENTURA, CELSO EFRAIN NOEL	4	20	Enlace
GUARDIA/ZENTENO, IGOR ALFRED	1	14	Enlace
HUAYPUNA/HUANCA, JOHANN FRANZ	4	20	Enlace
HUISA QUISPE, JOSE LUIS	4	20	Enlace
LEON/PAREDES, GUSTAVO MARTIN	1	14	Enlace
SONCCO/LUPA, JEAN CARLOS	3	18	Enlace
TAMO/TURPO, ERIKA JUDITH	2	18	Enlace
TORRES/LIMA, JOSE MANUEL	4	20	Enlace
VILLANUEVA/SANCHEZ, FERNANDO THOMAS	3	18	Enlace
VISA/FLORES, ALBERTO	1	14	Enlace



8. Evaluación Continua 1

Práctica 1

MSc. Vicente Machaca Arceda

30 de julio de 2020

DOCENTE	CARRERA	CURSO
MSc. Vicente Machaca Arceda	Escuela Profesional de Ciencia de la Computación	Computación Molecular Biológica

PRÁCTICA	TEMA	DURACIÓN
01	Secuenciamiento de ADN	3 horas

1. Competencias del curso

- Aplica las bases matemáticas y la teoría de la informática en algoritmos de Bioinformática.
- Analiza, diseña y propone soluciones frente a problemas bioinformáticos.
- Sabe cómo utilizar y conoce las bases computacionales de herramientas modernas de secuenciamiento, alineamiento, árboles filogenéticos y mapeo de genomas.

2. Competencias de la práctica

- Comprender las bases teórias del secuenciamiento de ADN.

3. Equipos y materiales

- Editor de texto Latex

4. Entregables

- Se debe elaborar un informe en Latex donde se desarrolle el trabajo solicitado.
- El informe se desarrollará en grupos de 4.
- El informe deberá estar correctamente citado utilizando las normas APA.

5. Desarrollo

1. Desarrolle un informe sobre secuenciamiento de ADN con la siguiente estructura:
 - Definición de secuenciamiento de ADN.
 - Historia del secuenciamiento de ADN (cite los paper donde se publicó cada técnica).
 - Método de Sanger.
 - Problemas actuales del secuenciamiento de ADN.
 - Conclusiones
 - Referencias

Práctica 2

MSc. Vicente Machaca Arceda

12 de mayo de 2020

DOCENTE	CARRERA	CURSO
MSc. Vicente Machaca Arceda	Escuela Profesional de Ciencia de la Computación	Computación Molecular Biológica

PRÁCTICA	TEMA	DURACIÓN
02	Secuenciamiento de ADN	3 horas

1. Competencias del curso

- Aplica las bases matemáticas y la teoría de la informática en algoritmos de Bioinformática.
- Analiza, diseña y propone soluciones frente a problemas bioinformáticos.
- Sabe cómo utilizar y conoce las bases computacionales de herramientas modernas de secuenciamiento, alineamiento, árboles filogenéticos y mapeo de genomas.

2. Competencias de la práctica

- Comprender los métodos utilizados en *Next Generation Sequence* (NGS)

3. Equipos y materiales

- Editor de texto Latex

4. Entregables

- Se debe elaborar un informe y una presentación en Latex donde se desarrolle el trabajo solicitado.
- El informe se desarrollará en grupos de 4.
- El informe deberá estar correctamente citado utilizando las normas APA.

5. Desarrollo

1. Escoga uno de los siguiente temas:

- Pyrosequencing (454 Life Sciences)
- Semiconductor sequencing (Ion Torrent).
- Reversible chain-termination sequencing (Illumina).
- Single-molecule sequencing (Pacific Biosciences and MinION)

2. Desarrolle un informe con la siguiente estructura como minimo:

- Introducción
- Descripción del método.
 - Pasos o etapas del método.
 - Longitud de fragmentos leidos.
- Ventajas y desventajas.
- Conclusiones
- Referencias

8.1 Evidencias

La evidencia de los Laboratorios realizados por los estudiantes en la Evaluación Continua 1 se muestran en la siguiente tabla:

Laboratorio	Evidencia
Laboratorio 1	Link
Laboratorio 2	Link

Tabla 8.1: Evidencia Evaluación Continua 1



9. Evaluación Continua 2

Práctica 3

MSc. Vicente Machaca Arceda

20 de agosto de 2020

DOCENTE	CARRERA	CURSO
MSc. Vicente Machaca Arceda	Escuela Profesional de Ciencias de la Computación	Computación molecular Biológica

PRÁCTICA	TEMA	DURACIÓN
01	Thresholding	3 horas

1. Competencias del curso

- Aplica las bases matemáticas y la teoría de la informática en algoritmos de Bioinformática.
- Analiza, diseña y propone soluciones frente a problemas bioinformáticos.
- Sabe cómo utilizar y conoce las bases computacionales de herramientas modernas de secuenciamiento, alineamiento, árboles filogenéticos y mapeo de genomas.

2. Competencias de la práctica

- Utilizar herramientas de Dot plot.
- Implementar el algoritmo Dot plot para alineamiento de secuencias.

3. Equipos y materiales

- Python
- Matplotlib
- Numpy
- BioPython
- Cuenta en Github

4. Entregables

- Se debe elaborar un informe en Latex donde se responda a cada ejercicio de la Sección 5.
- En el informe se debe agregar un enlace al repositorio Github donde esta el código.
- En el informe se debe agregar el código fuente así como capturas de pantalla de la ejecución y resultados del mismo.

5. Ejercicios

1. Descarge dos secuencias de proteínas (las utilizadas en clases) y ejecute el siguiente código. Comente sus resultados.

```
# dot_matrix.py
from Bio import SeqIO

sequences = SeqIO.parse("P21333.fasta", "fasta")
for record in sequences:
    data1 = str(record.seq.upper()) # the fasta file just have one sequence

sequences = SeqIO.parse("Q8BTM8.fasta", "fasta")
for record in sequences:
    data2 = str(record.seq.upper()) # the fasta file just have one sequence

print(data1)
print(data2)
```

2. Ahora usted debe implementar un programa que genere un Dot matrix. Se recomienda utilizar Matplot para la gráfica, de igual manera no es necesario dibujar las líneas, basta con dibujar los puntos por cada coincidencia (no olvide incluir el *windows size* y un *threshold*).
3. Descargue otras secuencias y genere el *Dot matrix*, evalúe sus resultados y comente.

Práctica 05

MSc. Vicente Machaca Arceda

16 de junio de 2020

DOCENTE	CARRERA	CURSO
MSc. Vicente Machaca Arceda	Escuela Profesional de Ciencia de la Computación	Computación Molecular Biológica

PRÁCTICA	TEMA	DURACIÓN
05	Alineamiento de Secuencias con Programación Dinámica	3 horas

1. Competencias del curso

- Aplica las bases matemáticas y la teoría de la informática en algoritmos de Bioinformática.
- Analiza, diseña y propone soluciones frente a problemas bioinformáticos.
- Sabe cómo utilizar y conoce las bases computacionales de herramientas modernas de secuenciamiento, alineamiento, árboles filogenéticos y mapeo de genomas.

2. Competencias de la práctica

- Aplica las bases matemáticas y la teoría de la informática en algoritmos de Alineamiento de Secuencias con Programación Dinámica.

3. Equipos y materiales

- Latex
- Conección a internet
- Python
- Matplotlib
- Numpy
- BioPython
- Cuenta en Github

4. Entregables

- Se debe elaborar un informe en Latex donde se responda a cada ejercicio de la Sección 5.
- En el informe se debe agregar un enlace al repositorio Github donde esta el código.
- En el informe se debe agregar el código fuente así como capturas de pantalla de la ejecución y resultados del mismo.

5. Ejercicios

- Implemente el algoritmo de alineamiento de secuencias utilizando programación dinámica (Needleman–Wunsch). Evalúe sus resultados con las secuencias:

- S_1 : AGC
- S_2 : AAG

Utilice $gapOpen = gapEXTEND = -5$ y la siguiente matriz de sustitución:

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

La salida debe incluir el *score matrix* y todos los alineamientos posibles.

- Utilice el algoritmo implementado anteriormente, pero esta vez con secuencias reales. Se recomienda utilizar las secuencias de proteínas de la Práctica 3. Utilice $gapOpen = gapEXTEND = -5$, $identicalMatch = 2$ y $mismatch = -2$
- Evalue la pregunta anterior con otros valores de $gapOpen$, $gapEXTEND$, $identicalMatch$ y $mismatch$. Verifique si obtiene el mismo alineamiento y comente sus resultados.

Práctica 04

MSc. Vicente Machaca Arceda

8 de junio de 2020

DOCENTE	CARRERA	CURSO
MSc. Vicente Machaca Arceda	Escuela Profesional de Ciencia de la Computación	Computación Molecular Biológica

PRÁCTICA	TEMA	DURACIÓN
04	Lecturas	3 horas

1. Competencias del curso

- Aplica las bases matemáticas y la teoría de la informática en algoritmos de Bioinformática.
- Analiza, diseña y propone soluciones frente a problemas bioinformáticos.
- Sabe cómo utilizar y conoce las bases computacionales de herramientas modernas de secuenciamiento, alineamiento, árboles filogenéticos y mapeo de genomas.

2. Competencias de la práctica

- Comprender algunos tópicos en mutaciones, historia de la bioinformática y *Virology*.

3. Equipos y materiales

- Latex
- Conección a internet

4. Entregables

- Se debe elaborar un informe donde se explique cada tema asignado.
- Exposición de cada tema, cada exposición no debe durar mas de 10 min.

5. Temas

A continuación se presentan los temas, cada grupo escogerá un tema y revisará el material de ayuda como mínimo.

1. Historia de la Bioinformática. Se debe explicar cada hito importante de la Bioinformática, sustentar cada evento con las publicaciones en revistas científicas. Material de ayuda:
 - Video
 - The Roots of Bioinformatics
2. Mutaciones. Material de ayuda:
 - DNA Replication and Causes of Mutation
 - Genetic Mutation
 - Mutations: Types and Causes
3. Replicación de virus. Material de ayuda:
 - Virus Replication
 - DNA Viruses in Eukaryotes
4. COVID-19. Material de ayuda:
 - The Human Coronavirus Disease COVID-19
 - COVID-19 Research registry

Práctica 05

MSc. Vicente Machaca Arceda

22 de junio de 2020

DOCENTE	CARRERA	CURSO
MSc. Vicente Machaca Arceda	Escuela Profesional de Ciencia de la Computación	Computación Molecular Biológica

PRÁCTICA	TEMA	DURACIÓN
05	Alineamiento de Secuencias con Programación Dinámica	3 horas

1. Competencias del curso

- Aplica las bases matemáticas y la teoría de la informática en algoritmos de Bioinformática.
- Analiza, diseña y propone soluciones frente a problemas bioinformáticos.
- Sabe cómo utilizar y conoce las bases computacionales de herramientas modernas de secuenciamiento, alineamiento, árboles filogenéticos y mapeo de genomas.

2. Competencias de la práctica

- Aplica las bases matemáticas y la teoría de la informática en algoritmos de Alineamiento de Secuencias con Programación Dinámica.

3. Equipos y materiales

- Latex
- Conección a internet
- Python
- Matplotlib
- Numpy
- BioPython
- Cuenta en Github

4. Entregables

- Se debe elaborar un informe en Latex donde se responda a cada ejercicio de la Sección 5.
- En el informe se debe agregar un enlace al repositorio Github donde esta el código.
- En el informe se debe agregar el código fuente así como capturas de pantalla de la ejecución y resultados del mismo.

5. Ejercicios

1. Encuentre el mejor alineamiento global entre las secuencias **AAAC** y **AGC**,, con el siguiente *scoring scheme*:
+1 for match, -1 for mismatch and -2 for an alignment with a gap.
2. Encuentre el mejor alineamiento global entre las secuencias **ATAG** y **TTCG**, con el siguiente *scoring scheme*:
+1 for match, -1 for mismatch and -1 for an alignment with a gap.
3. Encuentre el mejor alineamiento local entre las secuencias **ATACTGGG** y **TGACTGAG**,, con el siguiente *scoring scheme*: *+1 for match, -1 for mismatch and -2 for an alignment with a gap.*

Solución de la práctica 05

MSc. Vicente Machaca Arceda

4 de julio de 2020

DOCENTE	CARRERA	CURSO
MSc. Vicente Machaca Arceda	Escuela Profesional de Ciencia de la Computación	Computación Molecular Biológica

PRÁCTICA	TEMA	DURACIÓN
05	Alineamiento de Secuencias con Programación Dinámica	3 horas

1. Competencias del curso

- Aplica las bases matemáticas y la teoría de la informática en algoritmos de Bioinformática.
- Analiza, diseña y propone soluciones frente a problemas bioinformáticos.
- Sabe cómo utilizar y conoce las bases computacionales de herramientas modernas de secuenciamiento, alineamiento, árboles filogenéticos y mapeo de genomas.

2. Competencias de la práctica

- Aplica las bases matemáticas y la teoría de la informática en algoritmos de Alineamiento de Secuencias con Programación Dinámica.

3. Equipos y materiales

- Latex
- Conección a internet
- Python
- Matplotlib
- Numpy
- BioPython
- Cuenta en Github

4. Entregables

- Se debe elaborar un informe en Latex donde se responda a cada ejercicio de la Sección 5.
- En el informe se debe agregar un enlace al repositorio Github donde esta el código.
- En el informe se debe agregar el código fuente así como capturas de pantalla de la ejecución y resultados del mismo.

5. Ejercicios

1. Encuentre el mejor alineamiento global entre las secuencias **AAAC** y **AGC**,, con el siguiente *scoring scheme*:
+1 for match, -1 for mismatch and -2 for an alignment with a gap.

			A	G	C
		0	1	2	3
	0	0	← -2	← -4	← -6
A	1	↑-2	↖ 1	← -1	← -3
A	2	↑-4	↖ ↑ -1	↖ 0	↖ ← -2
A	3	↑-6	↖ ↑ -3	↖ ↑ -2	↖ -1
C	4	↑-8	↑ -5	↖ ↑ -4	↖ -1

Alignments	A A A C - A G C	A A A C A G - C	A A A C A - G C
Score	(-2)+1+(-1)+1 = -1	1+(-1)+(-2)+1 = -1	1+ (-2)+(-1)+1 = -1
Solution	Best Alignment	Best Alignment	Best Alignment

2. Encuentre el mejor alineamiento global entre las secuencias **ATAG** y **TTCG**, con el siguiente *scoring scheme*:
+1 for match, -1 for mismatch and -1 for an alignment with a gap.

			T	T	C	G
		0	1	2	3	4
	0	0	← -1	← -2	← -3	← -4
A	1	↑-1	↖ ← -1	↖ ← -2	↖ ← -3	↖ ← -4
T	2	↑-2	↖ 0	↖ 0	← -1	← -2
A	3	↑-3	↑ -1	↖ ↑ -1	↖ -1	↖ -2
G	4	↑-4	↑ -2	↖ ↑ -2	↖ ↑ -2	↖ 0

Alignments	A T A G T T C G	A - T A G - T T C G
Score	(-1)+1+(-1)+1 = 0	(-1)+(-1)+1-1+1 = -1

3. Encuentre el mejor alineamiento local entre las secuencias **ATACTGGG** y **TGACTGAG**,, con el siguiente scoring scheme: +1 for match, -1 for mismatch and -2 for an alignment with a gap.

			T	G	A	C	T	G	A	G
		0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0	0
A	1	0	0	0	↖1	0	0	0	↖1	0
T	2	0	↖1	0	0	↖1	↖1	0	0	↖0
A	3	0	0	↖0	↖1	0	↖0	↖0	↖1	0
C	4	0	0	0	0	↖2	←0	0	0	↖0
T	5	0	↖1	0	0	0	↖3	←1	0	0
G	6	0	0	↖2	←0	0	↑1	↖4	←2	↖1
G	7	0	0	↑0	0	0	0	↖2	↖3	↖3
G	8	0	0	↖1	0	0	0	↖1	↖	↖4 1↑

Alignments	A C T G G G A C T G A G	A C T G A C T G
Score	$1+1+1+1+(-1)+1 = 4$	$1+1+1+1 = 4$

9.1 Evidencias

La evidencia de los Laboratorios realizados por los estudiantes en la Evaluación Continua 2 se muestran en la siguiente tabla:

Laboratorio	Evidencia
Laboratorio 3	Link
Laboratorio 4	Link
Laboratorio 5	Link

Tabla 9.1: Evidencia Evaluación Continua 2



10. Evaluación Continua 3

Práctica 06

MSc. Vicente Machaca Arceda

22 de junio de 2020

DOCENTE	CARRERA	CURSO
MSc. Vicente Machaca Arceda	Escuela Profesional de Ciencia de la Computación	Computación Molecular Biológica

PRÁCTICA	TEMA	DURACIÓN
06	Alineamiento global de Secuencias con Programación Dinámica	3 horas

1. Competencias del curso

- Aplica las bases matemáticas y la teoría de la informática en algoritmos de Bioinformática.
- Analiza, diseña y propone soluciones frente a problemas bioinformáticos.
- Sabe cómo utilizar y conoce las bases computacionales de herramientas modernas de secuenciamiento, alineamiento, árboles filogenéticos y mapeo de genomas.

2. Competencias de la práctica

- Aplica las bases matemáticas y la teoría de la informática en algoritmos de Alineamiento global de Secuencias con Programación Dinámica.

3. Equipos y materiales

- Latex
- Conección a internet
- Python
- Matplotlib
- Numpy
- BioPython
- Cuenta en Github

4. Entregables

- Se debe elaborar un informe en Latex donde se responda a cada ejercicio de la Sección 5.
- En el informe se debe agregar un enlace al repositorio Github donde esta el código.
- En el informe se debe agregar el código fuente así como capturas de pantalla de la ejecución y resultados del mismo.

5. Ejercicios

- Implemente el algoritmo de alineamiento de secuencias utilizando programación dinámica (Needleman–Wunsch). Evalúe sus resultados con las secuencias:

- S_1 : AGC
- S_2 : AAG

Utilice $gapOpen = gapEXTEND = -5$ y la siguiente matriz de sustitución:

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

La salida debe incluir el *score matrix* y todos los alineamientos posibles.

- Utilice el algoritmo implementado anteriormente, pero esta vez con secuencias reales. Se recomienda utilizar las secuencias de proteínas de la Práctica 3. Utilice $gapOpen = gapEXTEND = -5$, $identicalMatch = 2$ y $mismatch = -2$
- Evalue la pregunta anterior con otros valores de $gapOpen$, $gapEXTEND$, $identicalMatch$ y $mismatch$. Verifique si obtiene el mismo alineamiento y comente sus resultados.

Práctica 07

MSc. Vicente Machaca Arceda

22 de junio de 2020

DOCENTE	CARRERA	CURSO
MSc. Vicente Machaca Arceda	Escuela Profesional de Ciencia de la Computación	Computación Molecular Biológica

PRÁCTICA	TEMA	DURACIÓN
07	Alineamiento local con programación dinámica	3 horas

1. Competencias del curso

- Aplica las bases matemáticas y la teoría de la informática en algoritmos de Bioinformática.
- Analiza, diseña y propone soluciones frente a problemas bioinformáticos.
- Sabe cómo utilizar y conoce las bases computacionales de herramientas modernas de secuenciamiento, alineamiento, árboles filogenéticos y mapeo de genomas.

2. Competencias de la práctica

- Aplica las bases matemáticas y la teoría de la informática en algoritmos de Alineamiento local con programación dinámica.

3. Equipos y materiales

- Latex
- Conección a internet
- Python
- Matplotlib
- Numpy
- BioPython
- Cuenta en Github

4. Entregables

- Se debe elaborar un informe en Latex donde se responda a cada ejercicio de la Sección 5.
- En el informe se debe agregar un enlace al repositorio Github donde esta el código.
- En el informe se debe agregar el código fuente así como capturas de pantalla de la ejecución y resultados del mismo.

5. Ejercicios

1. Implemente el algoritmo de alineamiento local utilizando el algoritmo de Smith-waterman. Evalúe sus resultados con las secuencias:

- S_1 : AGC
- S_2 : AAG

Utilice $gapOpen = gapEXTEND = -5$ y la siguiente matriz de sustitución:

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

La salida debe incluir el *score matrix* y todos los alineamientos posibles.

2. Utilice el algoritmo implementado anteriormente, pero esta vez con secuencias reales. Se recomienda utilizar las secuencias de ADN. Utilice $gapOpen = gapEXTEND = -5$, $identicalMatch = 2$ y $mismatch = -2$. Puede utilizar estas bases de datos:

- NCBI virus.
- Virus pathogen resource.
- Castor.

3. Evalúe la pregunta anterior con otros valores de $gapOpen$, $gapEXTEND$, $identicalMatch$ y $mismatch$. Verifique si obtiene el mismo alineamiento y comente sus resultados.

10.1 Evidencias

La evidencia de los Laboratorios realizados por los estudiantes en la Evaluación Continua 3 se muestran en la siguiente tabla:

Laboratorio	Evidencia
Laboratorio 6	Link
Laboratorio 7	Link

Tabla 10.1: Evidencia Evaluación Continua 3



11. Registro de Asistencia

Apellidos y Nombres	%
ARCOS/PONCE, SERGIO MANUEL	90.91
BERMUDEZ/NAVARRO, WILLIAN BRAULIO	81.82
CAYLLAHUE/CCORA, RENZO AUGUSTO	90.91
CHAMBI APAZA, SYOMIRA INES	100
CHAVEZ LOPEZ CAROLINA BONNIE	100
CONDORI/MANSILLA, WILLIAM SILVERIO	81.82
DEXTRE/AIQUIPA, MARKS CRISTOPHER	100
DIAZ/VENTURA, CELSO EFRAIN NOEL	90.91
GUARDIA/ZENTENO, IGOR ALFRED	72.73
HUAYPUNA/HUANCA, JOHANN FRANZ	100
HUISA QUISPE, JOSE LUIS	90.91
LEON/PAREDES, GUSTAVO MARTIN	90.91
SONCCO/LUPA, JEAN CARLOS	72.73
TAMO/TURPO, ERIKA JUDITH	100
TORRES/LIMA, JOSE MANUEL	100
VILLANUEVA/SANCHEZ, FERNANDO THOMAS	63.64
VISA/FLORES, ALBERTO	27.27



12. Registro de Notas

Apellidos y Nombres	EC1	EP1	EC2	EP2	EP 3	EC3	Nota
ARCOS/PONCE, SERGIO MANUEL	14	8	13	16	18	15	14
BERMUDEZ/NAVARRO, WILLIAN BRAULIO	12	8	6	0	0	0	4
CAYLLAHUE/CCORA, RENZO AUGUSTO	15	14	17	12	18	15	15
CHAMBI APAZA, SYOMIRA INES	14	12	17	13	18	18	16
CHAVEZ LOPEZ CAROLINA BONNIE	16	12	17	10	18	15	15
CONDORI/MANSILLA, WILLIAM SILVERIO	14	8	17	10	18	15	14
DEXTRE/AIQUIPA, MARKS CRISTOPHER	12	8	17	11	18	18	14
DIAZ/VENTURA, CELSO EFRAIN NOEL	15	12	9	6	20	15	13
GUARDIA/ZENTENO, IGOR ALFRED	13	10	13	14	14	15	13
HUAYPUNA/HUANCA, JOHANN FRANZ	15	4	9	7	20	15	12
HUISA QUISPE, JOSE LUIS	0	10	9	7	20	15	11
LEON/PAREDES, GUSTAVO MARTIN	12	10	13	12	14	15	13
SONCCO/LUPA, JEAN CARLOS	14	6	17	6	18	18	14
TAMO/TURPO, ERIKA JUDITH	15	16	17	6	18	15	15
TORRES/LIMA, JOSE MANUEL	17	16	19	18	20	15	18
VILLANUEVA/SANCHEZ, FERNANDO	16	8	17	12	18	18	15
VISA/FLORES, ALBERTO	13	8	13	14	14	15	13



13. Material del Curso

Las presentaciones utilizadas en el desarrollo del curso se encuentran en la siguiente tabla:

Tema	Material
Introducción	Link
Biología de la celula	Link
Tamaño y números	Link
Estructura y replicación del DNA	Link
Secuenciamiento de DNA	Link
Alineamiento	Link
Alineamiento - Programación dinámica	Link
Alineamiento - Alineamiento local	Link
BLAST	Link
Árboles filogenéticos	Link

Tabla 13.1: Material del Curso



14. Evaluación de desempeño

1. Nombre del Curso

Código	Nombre	Semestre
1005155	Computación Molecular Biológica	2020 A

2. Mapeo del Resultado del Estudiante

	[a]	[b]	[c]	[d]	[e]	[f]	[g]	[h]	[i]	[j]
CG	2	-	-	-	-	-	2	-	-	-

3. Planificación de la Evaluaciones

3.1. **Resultado del Estudiante:** (a) Conocimientos de Computación: La capacidad de aplicar conocimientos de matemáticas, ciencias, computación y una especialidad de computación apropiados para los resultados del estudiante y la disciplina del programa. Nivel 1 (Comprende)

Indicadores de Desempeño						
Indicadores de Desempeño	Curso	Métodos Assessment	Fuente de Assessment (curso, semana y actividad)	Ciclo de Assessment	Coordinador Assessment	Nivel de logro esperado
a1. Prácticas de laboratorio	CG	Rúbrica (Directa)	Todo el semestre Práctica 1 - 10	2020-A	Vicente Machaca Arceda	80%

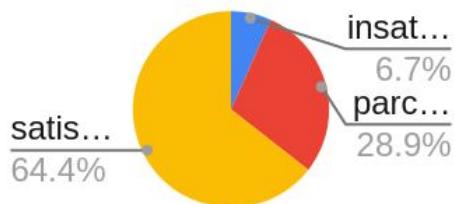
Rúbrica a Usarse				
Indicadores de Desempeño	Niveles del Logro			
	1. Insatisfactorio	2. Parcialmente Satisfactoria	3. Satisfactorio	4. Excelente
a1. Prácticas de laboratorio	No desarrolla las prácticas	Desarrolla las prácticas incompletas	Desarrolla todas las prácticas pero el código no es óptimo	Desarrolla todas las prácticas y el código es óptimo

Nivel del Logro de los Resultados del Estudiante para el Indicador		
Apellidos y Nombres	a1	Evidencia

ARCOS/PONCE, SERGIO MANUEL	3	Evidencia
BERMUDEZ/NAVARRO, WILLIAN BRAULIO	1	Evidencia
CAYLLAHUE/CCORA, RENZO AUGUSTO	3	Evidencia
CHAMBI APAZA, SYOMIRA INES	3	Evidencia
CHAVEZ LOPEZ CAROLINA BONNIE	3	Evidencia
CONDORI/MANSILLA, WILLIAM SILVERIO	3	Evidencia
DEXTRE/AIQUIPA, MARKS CRISTOPHER	3	Evidencia
DIAZ/VENTURA, CELSO EFRAIN NOEL	2	Evidencia
GUARDIA/ZENTENO, IGOR ALFRED	3	Evidencia
HUAYPUNA/HUANCA, JOHANN FRANZ	2	Evidencia
HUISA QUISPE, JOSE LUIS	2	Evidencia
LEON/PAREDES, GUSTAVO MARTIN	2	Evidencia
SONCCO/LUPA, JEAN CARLOS	2	Evidencia
TAMO/TURPO, ERIKA JUDITH	3	Evidencia
TORRES/LIMA, JOSE MANUEL	3	Evidencia
VILLANUEVA/SANCHEZ, FERNANDO THOMAS	3	Evidencia
VISA/FLORES, ALBERTO	3	Evidencia

Cuadro resumen											
Indicadores	Insatisfactorio		parcialmente satisfactorio		satisfactorio		excelente		NIVEL DE LOGRO		
	#	%	#	%	#	%	#	%	#	%	OBJETIVO
a1	1	3	5	13	11	29	0	0	11	29	1

a1



- 3.2. Resultado del Estudiante:** (g) Aprendizaje Continuo: La capacidad de reconocer la necesidad del aprendizaje y el desarrollo profesional continuo. Nivel 2 (Aplica en un nivel intermedio)

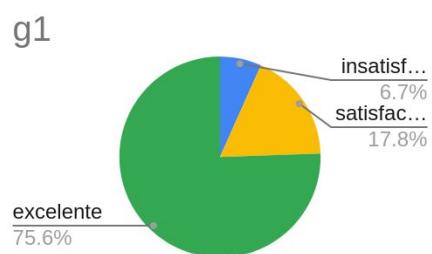
Indicadores de Desempeño						
Indicadores de Desempeño	Curso	Métodos Assessment	Fuente de Assessment (curso, semana y actividad)	Ciclo de Assessment	Coordinador Assessment	Nivel de logro esperado
g1. Trabajo de implementación de un paper	CG	Rúbrica (Directa)	Todo el semestre EP3	2020-A	Vicente Machaca Arceda	80%

Rúbrica a Usarse						
Indicadores de Desempeño	Niveles del Logro					
	1. Insatisfactorio	2. Parcialmente Satisfactoria	3. Satisfactorio	4. Excelente		
g1. Trabajo de investigación sobre filtros en imágenes	No desarrolla el trabajo	Casi no desarrolla el trabajo	Desarrolla el trabajo con observaciones	Desarrolla el trabajo a la perfección		

Nivel del Logro de los Resultados del Estudiante para el Indicador			
Apellidos y Nombres	g1	Evidencia	
ARCOS/PONCE, SERGIO MANUEL	4	Evidencia	
BERMUDEZ/NAVARRO, WILLIAN BRAULIO	0	Evidencia	
CAYLLAHUE/CCORA, RENZO AUGUSTO	4	Evidencia	
CHAMBI APAZA, SYOMIRA INES	4	Evidencia	
CHAVEZ LOPEZ CAROLINA BONNIE	4	Evidencia	
CONDORI/MANSILLA, WILLIAM SILVERIO	4	Evidencia	
DEXTRE/AIQUIPA, MARKS CRISTOPHER	4	Evidencia	
DIAZ/VENTURA, CELSO EFRAIN NOEL	4	Evidencia	
GUARDIA/ZENTENO, IGOR ALFRED	3	Evidencia	
HUAYPUNA/HUANCA, JOHANN FRANZ	4	Evidencia	
HUISA QUISPE, JOSE LUIS	4	Evidencia	

LEON/PAREDES, GUSTAVO MARTIN	3	Evidencia
SONCCO/LUPA, JEAN CARLOS	4	Evidencia
TAMO/TURPO, ERIKA JUDITH	4	Evidencia
TORRES/LIMA, JOSE MANUEL	4	Evidencia
VILLANUEVA/SANCHEZ, FERNANDO THOMAS	4	Evidencia
VISA/FLORES, ALBERTO	3	Evidencia

Cuadro resumen											
Indicadores	Insatisfactorio		parcialmente satisfactorio		satisfactorio		excelente		NIVEL DE LOGRO		
	#	%	#	%	#	%	#	%	#	%	OBJETIVO
g1	1	3	0	0	3	8	13	34	16	42	80





15. Informe Final del Curso

 <p>UNSA UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA</p>	<p>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS UNIDAD DE CALIDAD</p>	Código: FO-02
		Versión: 01-2018/12/11
	FORMATO: INFORME FINAL DEL CURSO	Página 1 de 3

Escuela Profesional	Fecha
Ciencia de la Computación	02-09-2020

Código	Nombre de la Asignatura	Grupo	Tipo (marcar recuadro)					
1005155	Computación Molecular Biológica	A	<input type="checkbox"/> Teor.	<input type="checkbox"/> 2	<input type="checkbox"/> Práct.	<input type="checkbox"/> 2	<input type="checkbox"/> Lab.	<input type="checkbox"/> 2

Apellidos y Nombres del Docente

Vicente Enrique Machaca Arceda

E-mail personal	vmachacaa@unsa.edu.pe	Teléfono	992578385
-----------------	-----------------------	----------	-----------

DEL CURSO

(%)	Porcentaje de cumplimiento del sílabo	80
Nº	Prácticas calificadas realizadas	8
Nº	Experiencias de laboratorio realizadas	-
Nº	Proyectos y trabajos de investigación formativa realizados	1
Nº	Estudiantes matriculados	17
Nº	Estudiantes aprobados	16
Nº	Estudiantes desaprobados	1
Nº	Estudiantes que no se presentaron (NSP)	0
	Nota final más alta	18
	Nota final promedio	13
	Nota final más baja	4

Observaciones:

- De los estudiantes.** Nivel académico, conocimientos previos, interés en el curso, estilos de aprendizaje.

- Los alumnos aprenden mejor realizando prácticas de laboratorio.
- Algunos alumnos se descuidaron en el primer examen

2. Asistencia y puntualidad de los estudiantes.

La asistencia en promedio supera el 90% de alumnos, y de ellos una puntualidad mayor al 80%.

3. Del sílabo. ¿Qué temas del sílabo no se han completado? ¿Considera todos los temas adecuados? ¿Qué temas requieren más tiempo de dedicación?

- No se completaron algunos temas de tópicos en Bioinformática, debido a la Pandemia y problemas de salud

4. Uso del Aula Virtual. Información colocada y cantidad de visitas estudiantiles.

En ese semestre se ha utilizado el aula virtual.

5. Administrativas. Disponibilidad de recursos en aulas y laboratorios.

- Las clases fueron virtuales

6. Sílabo por competencias. ¿Qué actividades realiza usted para la evaluación del cumplimiento de las competencias de su curso, por los estudiantes? ¿Podría presentar una propuesta de evaluación de las competencias de su curso?

Se realizaron prácticas y un trabajo final

7. Mejora continua. ¿Qué actividades de mejora continua ha realizado en el curso en este ciclo? y ¿Qué actividades propone para el siguiente semestre, para mejorar el rendimiento académico del estudiante?

- Se ha tenido todo el material en Latex y con una plantilla brindada por la EPP
- Todas las sesiones han sido grabadas

8. Actualización docente. Ha seguido usted algún curso de actualización profesional o docente en los últimos dos años. Indique el tema del curso y su duración.

- De ha seguido varios cursos en línea sobre Bioinformática y además se ha publicado en revistas indizadas, investigaciones en el área de Bioinformática

9. Comentarios y Recomendaciones. Serán útiles para mejorar el Plan de Estudios y los servicios que brindan la Escuela, la Facultad y la Universidad.

- Proponer el desarrollo de un sistema para el control de los portafolios.



MSc. Ing. Vicente Machaca Arceda

NOTA:

Enviar el informe completo a la dirección electrónica de la escuela profesional del curso

Remitir copia a la siguiente dirección: fips_ucaf@unsa.edu.pe

Imprimir y entregarlo firmado a la secretaría de la Escuela Profesional correspondiente.