CRISP-DM

Sistema de recomendación piloto para programación de clases del departamento de Ingeniería de Sistemas en la Pontificia Universidad Javeriana.

EDWARD ANDRÉS CORTÉS PEÑA
PABLO MIGUEL NUÑEZ GONZÁLEZ
SARA ISABELA VERGARA AGUILAR

BOGOTÁ D.C.
2021

Documento CRISP-DM

Información del documento

Proyecto	SRDIS
Identificador	SRDIS _CRISP-DM_02122020
Nombre del Documento	CRISP-DM
Estado	Finalizado
Tipo de Documento	Mediano
Etapa	Implementación Modelos Analíticos
Responsable:	Edward Cortés

Control de versiones del documento

Versión	Fecha	Descripción del cambio	Elaborado por	Autorizado por
0.1	15/06/2021	Creación de documento	Edward Cortes	
1.0	28/09/2021	Sección verificación	Todo el equipo	

Contenido

LISTA DE TABLAS

LISTA DE FIGURAS

1. Conocimiento del negocio

1.1 Contextualización de Industria

Para el marco de desarrollo en este proyecto de grado se enfocó en el sector de educación a nivel de pregrado de la facultad de ingeniería de sistemas de la Pontificia Universidad Javeriana, para dar mayor contexto del sector que se está abordando se identifican los actores que están directamente interesados; estos actores son: a). La oficina de Admisiones y Registro la cual es el ente encargado de brindar soporte técnico y funcional al proceso de programación de clases y catálogo de asignaturas para cada periodo académico. b). Departamento de Ingeniería de Sistemas en el grado académico de pregrado, es el encargado de registrar la programación de clases conforme a las solicitudes del Programa de Ingeniería de Sistemas y siendo este el ultimo actor interesado se identifica como el ente encargado de gestionar y brindar la información necesaria para programar los cursos que se van a ofertar en cada periodo. Actualmente cada uno de los departamentos de la universidad tienen la libertad de programar las clases necesarias para cumplir los planes académicos establecidos.

El Programa de Ingeniería de Sistemas expresa que el proceso de programación de clases es una tarea compleja, ya que hay variables o factores externos que impiden la planeación de manera adecuada; actualmente se cuenta dos (2) planes de estudio: uno antiguo que consta de 10 semestres y uno nuevo que consta de 8 semestres para pregrado.

Como estrategia para planear las asignaturas que se van a ofertar, el Programa de Ingeniería de Sistemas realiza dos encuestas:

- La primera encuesta tiene por objetivo conocer las asignaturas que los estudiantes planean inscribir en el siguiente semestre y con esto hacer una base de datos fija de cuantos estudiantes se van a tener por grupo para la programación, es importante resaltar que no tuvo una buena acogida, ya que solamente el 40 % de los estudiantes la diligencio.
- La segunda encuesta se realiza después de la primera cita de inscripciones para conocer los problemas que tuvieron los estudiantes con las asignaturas que quedaron pendientes por matricular y la demanda desconocida para la apertura de cupos necesarios.

Con base en la primera encuesta el programa realiza la solicitud de clases al Departamento de Ingeniería de Sistemas, debido a que ellos son los encargados de realizar el registro de la programación de clases en el sistema de información (plataforma PeopleSoft y de consultar la disponibilidad de profesores para las clases planeadas). Cuando el departamento ya cuenta con toda la programación registrada, se pasa al área de admisiones y registro para la asignación de los espacios.

Posteriormente se solicita a la oficina de admisiones la apertura de nuevos grupos o el ajuste en la capacidad de inscripción de las clases programadas, para la planeación se tiene un supuesto de estudiantes que se van a matricular y esto puede cambiar de un periodo a otro, dependiendo de si los estudiantes son o no de primer semestre, y que a partir del segundo semestre en adelante las asignaturas empiezan a tener requisitos.

Por lo anterior, el Programa de Ingeniería de Sistemas expresa la necesidad contar con un sistema automatizado de recomendaciones con el cual se facilite y sea más asertiva la planeación de asignaturas, capacidad de estudiantes por curso y posteriormente fijar las franjas horarias de cada una.

1.2 Objetivos de Negocio.

Establecer una estrategia que permita la identificación de asignaturas con mayor demanda y a partir de los datos históricos obtenidos, se genere un sistema de recomendaciones que permita optimizar el proceso de programación de clases desde el Departamento de Ingeniería de Sistemas en el grado académico de pregrado.

1.3 Objetivos De Minería de Datos.

- Identificar las variables más relevantes para la generación del modelo usando los datos entregados por el área encargada.
- Experimentar, validar y probar con varios modelos de algoritmos analíticos, para obtener el modelo óptimo que genere la recomendación que mejor explique la variable objetivo (Número de inscritos por asignatura para un periodo especifico) en el desarrollo de este proyecto.
- Implementar un sistema de visualización de resultados de recomendación, que pueda ser usado por el área encargada.

2. Entendimiento de los Datos

2.1 Datos iniciales

Los datos fueron consolidados y obtenidos por el departamento, la faculta de ingeniería de sistemas y el departamento de admisiones y registro; el cual fue entregado en un archivo con un formato de tipo (xlsx) y nombrado “ProgramaciónClases1510-2130.xlsx” el cual contiene 4354 datos que esta distribuidos en una serie de hojas con los datos de cada periodo determinado y cursado

- 1510: Esta hoja de la data set contiene los datos del año 2015 y el periodo 10 que inicio en el mes de febrero y termina en el mes de julio de cada año, la hoja contiene (datos 281, variables 43).
- 1530: Esta hoja de la data set contiene los datos del año 2015 y el periodo 30 que inicio en el mes de agosto y termina en el mes de noviembre de cada año, la hoja contiene (datos 287, variables 43).
- 1610: Esta hoja de la data set contiene los datos del año 2016 y el periodo 10 que inicio en el mes de febrero y termina en el mes de julio de cada año, la hoja contiene (datos 313, variables 43).
- 1630: Esta hoja de la data set contiene los datos del año 2016 y el periodo 30 que inicio en el mes de agosto y termina en el mes de noviembre de cada año, la hoja contiene (datos 309, variables 43).

- 1710: Esta hoja de la data set contiene los datos del año 2017 y el periodo 10 que inicio en el mes de febrero y termina en el mes de julio de cada año, la hoja contiene (datos 325, variables 43).
- 1730: Esta hoja de la data set contiene los datos del año 2017 y el periodo 30 que inicio en el mes de agosto y termina en el mes de noviembre de cada año, la hoja contiene (datos 298, variables 43).
- 1810: Esta hoja de la data set contiene los datos del año 2018 y el periodo 10 que inicio en el mes de febrero y termina en el mes de julio de cada año, la hoja contiene (datos 287, variables 43).
- 1830: Esta hoja de la data set contiene los datos del año 2018 y el periodo 30 que inicio en el mes de agosto y termina en el mes de noviembre de cada año, la hoja contiene (datos 309, variables 43).
- 1910: Esta hoja de la data set contiene los datos del año 2019 y el periodo 10 que inicio en el mes de febrero y termina en el mes de julio de cada año, la hoja contiene (datos 273, variables 43).
- 1930: Esta hoja de la data set contiene los datos del año 2019 y el periodo 30 que inicio en el mes de agosto y termina en el mes de noviembre de cada año, la hoja contiene (datos 283, variables 43).
- 2010: Esta hoja de la data set contiene los datos del año 2020 y el periodo 10 que inicio en el mes de febrero y termina en el mes de julio de cada año, la hoja contiene (datos 313, variables 43).
- 2030: Esta hoja de la data set contiene los datos del año 2020 y el periodo 30 que inicio en el mes de agosto y termina en el mes de noviembre de cada año, la hoja contiene (datos 274, variables 43).
- 2110: Esta hoja de la data set contiene los datos del año 2021 y el periodo 10 que inicio en el mes de febrero y termina en el mes de julio de cada año, la hoja contiene (datos 380, variables 43).
- 2130: Esta hoja de la data set contiene los datos del año 2021 y el periodo 30 que inicio en el mes de agosto y termina en el mes de noviembre de cada año, la hoja contiene (datos 422, variables 43).

2.2 Descripción de los datos

Variable	Tipo	Descripción
Grado	Varchar	Identificador para postgrado y pregrado (PREG/GRAD)
Descr	Varchar	Descripción de la asignatura del pensul
Catálogo	Integer	
Sesión	Integer	
ID Curso	Integer	Identificador del curso de la asignatura
Sección	Integer	Cantidad de sesiones para cada asignatura
Nº Clase	Integer	Identificador de la clase de la asignatura
ID Instal	Varchar	Identificación de instalación o asignación de salon
F Inicial	datetime64[ns]	Fecha de inicio del curso y clase
Fecha Final	datetime64[ns]	Fecha de finalización del curso y clase
Ciclo	Integer	Identificador del ciclo al que corresponde la asignatura

Org Acad	Varchar	Definición del departamento al que le corresponde la asignatura
Estado Clase	Varchar	Define el estado de la asignatura y tiene dos estados (Activo /sesión cancelada)
Tipo Clase	Varchar	Hace referencia si eta inscrita o sin inscripción
Componente	Varchar	Definición de la forma cómo se puede tomar la clase para cada estudiante
Gp Acad	Varchar	Campo que define el grupo académico que lo va a inscribir la asignatura
Modalidad	Varchar	Campo que define el tipo de asistencia a la clase
Nro Horas	float64	Campo que define el número de horas para la asignatura por día
Número Semanas	int64	Campo que define el número de semanas para la asignatura
Horas Semanales	int64	Campo que define la multiplicación del campo de Nro Horas * Números de semanas
Lunes	object	Campo determinado para el día especifico de la semana para tomar la a asignatura
Martes	object	Campo determinado para el día especifico de la semana para tomar la a asignatura
Miércoles	object	Campo determinado para el día especifico de la semana para tomar la a asignatura
Jueves	object	Campo determinado para el día especifico de la semana para tomar la a asignatura
Viernes	object	Campo determinado para el día especifico de la semana para tomar la a asignatura
Sábado	object	Campo determinado para el día especifico de la semana para tomar la a asignatura
Domingo	object	Campo determinado para el día especifico de la semana para tomar la a asignatura
Acceso	object	
Fecha Novedad	datetime64[ns]	Campo definido para fechas de cambios de horas o días de la asignatura
Estado de Asociación	object	Campo estado de la asignación de las asignaturas
Total Inscripciones	int64	Campo con el total de inscritos a la asignatura

Tabla 1. Descripción de variables de datos

2.3 Exploración de los datos

A partir del análisis invariada se obtienen los siguientes datos:

- **Grado:** Es la variable objetivo y se encuentra definida por 2 categorías 1= PREG, 2= GRAD.
- **Descr:** Es una variable categórica, que Describe el tipo de asignatura del

pensul para la carrrera. aplica la identificación de datos.

- **Catálogo:** Es una variable Integer, que Describe el tipo
- **Sesión:** Es una variable Integer, que Describe el tipo
- **ID Curso** Es una variable Integer, que Describe índice representativo de la clase definida
- **Sección:** Es una variable Integer, que Describe el tipo
- **Nº Clase:** Es una variable Integer, que Describe el código asignado a cada la clase de la del pemsul para la carrera
- **ID Instal:** Es una variable Integer, que Describe el código asignado a cada aula definida física de la universidad
- **F Inicial:** Es una variable data time, que Describe la fecha inicial en el periodo especifico de la clase de una asignatura
- **Fecha Final:** Es una variable data time, que Describe la fecha final en el periodo definido de la clase de una asignatura
- **Ciclo:** Es una variable Integer, que Describe el código definido para cada periodo a cursar
- **Org Acad:** Es una variable categórica, que Describe el departamento de la facultad que tiene a cargo las asignaturas
- **Estado Clase:** Es una variable categórica que identifica el estado de La clase y tiene 2 categorías 1: Activa 2: Sección Cancelada
- **Tipo Clase:** Es una variable categórica que identifica si la clase está inscrita o no, tiene 2 categorías 1: Sección Inscripción, 2: Sección Inscripción
- **Componente:** Es una variable categórica que identifica si la clase asistencia o actividad tiene 8 categorías 1: Laboratorio, 2: Práctico, 3, Proyecto, 4: Salida de Campo, 5: Seminario, 6: Teórico, 7: Teórico Práctico, 8: Trabajo de Campo.
- **Gp Acad:** Es una variable categórica que identifica el grado a académico de la facultad.
- **Modalidad:** Es una variable categórica que identifica la modalidad de profesor a signado a la asistencia tiene 2 categorías 1: Catedra, 2: Docencia.
- **Nº Horas:** Es una variable Integer que identifica el número de horas por clase
- **Número Semanas** Es una variable Integre que identifica el número de semanas por clase
- **Horas Semanales:** Es una variable Integer que identifica el número de horas por clase por semana
- **Lunes:** Es una variable categórica si la clase es tomada este día las categorías de finidas son 1: Y, 2: N
- **Martes:** Es una variable categórica si la clase es tomada este día las categorías de finidas son 1: Y, 2: N
- **Miércoles:** Es una variable categórica si la clase es tomada este día las categorías de finidas son 1: Y, 2: N
- **Jueves** Es una variable categórica si la clase es tomada este día las categorías de finidas son 1: Y, 2: N
- **Viernes:** Es una variable categórica si la clase es tomada este día las categorías de finidas son 1: Y, 2: N
- **Sábado:** Es una variable categórica si la clase es tomada este día las categorías de finidas son 1: Y, 2: N
- **Domingo:** Es una variable categórica si la clase es tomada este día las categorías de finidas son 1: Y, 2: N
- **Acceso:** Es una variable de tipo varchar es cual dice que por contabilizar

- **Fecha Novedad:** Es una variable de tipo datatime que identifica cambios efectuados en las fechas de las semanas de las clases
- **Estado de Asociación:** Es una variable categórica que identifica si la clase está en alguna de las siguientes categorías 1: Activas, 2: Inactivas
- **Total Inscripciones:** Es una variable integer que identifica el totalizado de estudiantes inscritos a la asignatura

2.4 Calidad de Datos

Se realiza la verificación de datos nulos en el data set y se encuentra que las variables Id Instal contiene 16 nulos, modalidad contiene 56 nulos, Acceso contiene 142 nulos, fecha de novedad contiene 8164 nulos; para las variables Doc, No contrato, Estado del contrato, Modalidad no se toman en cuenta para el análisis ya que son datos sensibles

```
In [93]: data0.isna().sum()

Out[93]: Grado                        0
         Descr                        0
         Catálogo                     0
         Sesión                       0
         ID Curso                     0
         Sección                      0
         Nº Clase                     0
         ID Instal                   16
         F Inicial                    0
         Fecha Final                  0
         Ciclo                        0
         Org Acad                     0
         Estado Clase                 0
         Tipo Clase                   0
         Componente                   0
         Gp Acad                      0
         Modalidad                   56
         ID                           0
         Doc ID                       2
         Nombre                       0
         Nº Contrato                 86
         Estado Contrato             56
         Modalidad.1                 56
         F Inicial Asociación         0
         F Final Asociación           0
         Hora Inicio Asociación       0
         Hora Fin Asociación          0
         Nro Horas                    0
         Número Semanas               0
         Horas Semanales              0
         Lunes                        0
         Martes                       0
         Miércoles                    0
         Jueves                       0
         Viernes                      0
         Sábado                       0
         Domingo                      0
         Acceso                     142
         Fecha Novedad             8164
         Estado de Asociación         0
         Total Inscripciones          0
         Nº Modelo Clase              0
         Rol Profesor                 0
         dtype: int64
```

Figura 1. Reconocimiento de datos.

Se realiza la validación de duplicados en el conjunto de datos y se identifica que tiene registros duplicados, es importante señalar que los valores duplicados tienen dependencia a variables categóricas que pueden tener de 2 a tres clases.

```
In [97]: data0[data0.duplicated()].sum()

Out[97]: Grado                 PREGPREGPREGPREGPREGPREGPREGPREGPREGPREGPR...
         Descr                 Análisis de AlgoritmosAnálisis de AlgoritmosAd...
         Catálogo                                                     1348800
         Sesión                111111111111111111111111111111111111111111111...
         ID Curso                                                     3761440
         Sección               111122335566771112233445577881010111211221122 3...
         Nº Clase                                                      821127
         Ciclo                                                         424310
         Org Acad              DPT-ISISTDPT-ISISTDPT-ISISTDPT-ISISTDPT-ISISTD...
         Estado Clase          ActivoActivoActivoActivoActivoActivoActivoActi...
         Tipo Clase            Sección InscripciónSección InscripciónSección ...
         Componente            Teorico PrácticoTeorico PrácticoTeorico Prácti...
         Gp Acad               INGENINGENINGENINGENINGENINGENINGENINGENINGENI...
         ID                                                       3248129463
         Doc ID                                                 3.62824e+10
         Nombre                Jaque Piraban,Robinson AndrésJaque Piraban,Rob...
         Nº Contrato                                             1.5068e+08
         Nro Horas                                                      578
         Número Semanas                                                4521
         Horas Semanales                                               8877
         Lunes                 NNNNNNNNNYNNNNNNNYNNNYNNNYNNYNNNNNNNNNNNYNNYNN...
         Martes                NNNYNNNYNNNYNYNNYNNYNNNYNNNNNNNYNNNYNYNYNNNYNYN...
         Miércoles             NYNNNYNNNNNNNNYNNNYNNNYNNNYYNNNNNNNNNNNNYNNNNN...
         Jueves                NNYNNNYNYNYNYNNYNNNNYNNNYNNNNYNNNYNYNYNNNYNNNY...
         Viernes               YNNNYNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...
         Sábado                NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNYNNNNNNNNNNNNN...
         Domingo               NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...
         Acceso                ContabilizarContabilizarContabilizarContabiliz...
         Estado de Asociación  InactivoInactivoInactivoInactivoInactivoInacti...
         Total Inscripciones                                             4282
         Nº Modelo Clase                                                  394
         Rol Profesor          ProfesorProfesorProfesorProfesorProfesorProfes...
         dtype: object
```

Figura 2. Comprobación de Registros duplicados.

3. Preparación de Datos

3.1 Selección de Datos

Es necesario disponer de todas las variables relacionadas previamente para la construcción de los diferentes modelos, adicional a esto se introducen otras variables producto de la transformación e integración de los datos, estos datos tienen como fin enriquecer la precisión del modelo aumentando directamente el coeficiente de correlación.

3.2.Limpieza de Datos y Construcción de Datos

Antes de iniciar la construcción del modelo fue necesario revisar la integridad de los datos, en primer lugar y dado que en su mayoría se obtienen de documentos de Excel, se dispuso del uso del lenguaje de programación Python además de las librerías "Pandas" y "sklearn.impute" para revisar:

- Datos faltantes
- Numerización de variables
- Métodos de Imputación de Datos
- Tipos de Datos

Se cargan los datos de la programación de clases, desde el periodo 1530 hasta el periodo 2130, para ello se usa la siguiente sentencia de Python.

```
[3]  #cargamos los datos de entrada
     data = pd.read_excel("/content/drive/MyDrive/ProgramaciónClases1510-2130.xlsx", sheet_name=None)
```

Figura 3. Cargue de datos de programación de clases.

Debido a que esta información se encuentra registrada en hojas de cálculo (una por cada periodo académico) se usa la sentencia "Sheet_name=None", la cual permite crear un dataframe para cada periodo.

```
#veamos cuantas dimensiones y registros contiene
data.keys()
```

```
odict_keys(['1510', '1530', '1610', '1630', '1710', '1730', '1810', '1830',
```

Figura 4. Cargue de datos programación de clases, dimensiones y registros.

Inicialmente se crean DataFrames independientes para cada periodo, es necesario concatenarlos para tener un único conjunto de datos sobre el cual se realicen consultas y operaciones, para ello se usa la instrucción "pd.concat" posteriormente se reinician los índices del DataFrame "df" para tener un conjunto íntegro y usable.

```
df = pd.DataFrame()
for name in data.keys():
    print(name)
    dat = pd.read_excel(r"C:\ProgramaciónClases1510-2130.xlsx", sheet_name=name)
    df=pd.concat([dat, df])
df.reset_index(level=0, inplace=True)
df.reset_index(level=0, inplace=True)
df.set_index('level_0')
df=df.drop(['index'],1)
```

```
1510
1530
1610
1630
1710
```

Figura 5. Cargue de datos modificados, datos íntegros y usables.

Los datos conseguidos tienen la siguiente estructura:

```
In [5]: df.head()
```

Out[5]:

	level_0	Grado	Descr	Catálogo	Sesión	ID Curso	Sección	Nº Clase	ID Instal	F Inicial	...	Jueves	Viernes	Sábado	Domingo	Acceso	Fecha Novedad	Est Aso
0	0	PREG	Administración Básica Linux	4800	1	4181	1	5721	NO-SALON	2021-07-19	...	N	N	N	N	Contabilizar	NaT	
1	1	PREG	Admón. Sistemas de Información	4800	1	4056	1	2549	NO-SALON	2021-07-19	...	Y	N	N	N	Contabilizar	NaT	
2	2	PREG	Admón. Sistemas de Información	4800	1	4056	1	2549	NO-SALON	2021-07-19	...	N	N	N	N	Contabilizar	NaT	
3	3	PREG	Admón. Sistemas de Información	4800	1	4056	2	2550	NO-SALON	2021-07-19	...	Y	N	N	N	Contabilizar	NaT	
4	4	PREG	Admón. Sistemas de Información	4800	1	4056	2	2550	NO-SALON	2021-07-19	...	N	N	N	N	Contabilizar	NaT	

Figura 6. Estructura de los datos cargados.

Posteriormente y con el ánimo de facilitar el uso y las consultas sobre el Dataframe se cambian los nombres de las variables, de la siguiente manera:

```
In [7]: # Renombramos las Columnas del DataFrame
df.set_axis(['index','Grado', 'Descr', 'Catalogo','Sesion','IDCurso','Seccion','Clase','IdInstal','FechaInicial','FechaFinal','C
             'OrgAcad','EstadoClase','TipoClase','Componente','GpAcad','Modalidad','ID','DocID','Nombre','NroContrato','Estado
             'IdModalidad','FechaInicialAsociación','FechaFinalAsociación','HoraInicioAsociacion','HoraFinAsociacion','NroHora
             'HorasSemanales', 'Lunes','Martes','Miercoles','Jueves','Viernes','Sabado','Domingo','Acceso','FechaNovedad',
              'EstadoAsociación', 'TotalInscripciones', 'NroModeloClase','RolProfesor'],
                 axis='columns', inplace=True)
```

Figura 7. Cambio de nombre de las variables

Se revisan los tipos de datos de cada una de las variables para tener una visión global de las operaciones que se deben realizar para estandarizar el conjunto de datos, es importante para este punto señalar que los algoritmos previstos usan únicamente datos numéricos para la construcción del modelo, por lo cual más adelante se construye una vista minable y se implementan técnicas de numerización de variables.

```
In [9]: #df.describe()
        df.dtypes

Out[9]: index                          int64
        Grado                         object
        Descr                         object
        Catalogo                      object
        Sesion                        object
        IDCurso                       object
        Seccion                       object
        Clase                          int64
        IdInstal                      object
        FechaInicial          datetime64[ns]
        FechaFinal            datetime64[ns]
        Ciclo                         object
        OrgAcad                       object
        EstadoClase                   object
```

Figura 8. Tipos de Datos

Para enriquecer la data se cargaron datos adicionales relacionados con el número de estudiantes que retiran o pierden una clase para un periodo académico especifico, estos datos toman el nombre de “Retiros o Perdidas”, a continuación, se describen las sentencias usadas para integrar esta información al conjunto de datos:

```
#cargamos los datos de los Retiros ls cuc
Retiros = pd.read_excel(r"C:\Users\cortes
Retiros["Periodo"]=1610
Retiros_1 = pd.read_excel(r"C:\Users\cort
Retiros_1["Periodo"]=1630
```

Figura 9. Asignaturas retiradas.

Estos datos se cargaron a un DataFrame individual y posteriormente se concatenaron para formar un DF llamado “Retiros”. Es importante señalar que los documentos de Excel que se usaron para crear el dataframe “Retiros” se diligencian de forma manual por parte del departamento de Ingeniería de Sistemas, con base en las solicitudes de retiro realizadas por los estudiantes en fechas específicas, por tanto, los archivos cargados no poseen una estructura definida, algunos de los documentos tienen variables diferentes o la definición del nombre de las columnas cambia de un archivo a otro.

Debido a que los documentos se estructuran de forma diferente, es necesario revisar las variables y si hay datos faltantes, el resultado es el siguiente:

```
# Limpieza de datos -- parte 3: Reemplazo/Imputación de datos faltantes

pd.DataFrame(Retiros).isna().sum() # Chequeamos si hay datos faltantes
```

```
ASIGNATURA              0
BECA                  370
BECARIO              1646
Cosecutivo              0
Créditos              163
ESTADO ACADEMICO       84
FECHA DE SOLICITUD      0
ID ASIGNATURA           0
MOTIVO                  0
NUMERO DE CLASE         1
Periodo                 0
Unnamed: 10          2015
Unnamed: 9           2016
dtype: int64
```

Figura 10. Limpieza de datos, df Datos Faltantes, df Retiros

Este dataframe contiene información relacionada con las solicitudes de retiro de asignaturas inscritas por parte de estudiantes para un periodo académico especifico, de allí se destacan dos variables: Id Asignatura y Periodo, estos dos datos se usan como llave primaria entre los dataframe “Retiros” y “df”, es importante destacar que estas variables no tienen datos faltantes.

Con el ánimo de habilitar las consultas se formatean los siguientes campos (Periodo – Id Asignatura), se asigna el tipo de dato “Object”.

```
[15] Retiros['Periodo'] = Retiros['Periodo'].astype('object')
     Retiros['ID ASIGNATURA'] = Retiros['ID ASIGNATURA'].astype('object')
```

Figura 11. Asignación tipos de datos

Debido a que el dataframe ‘df’ contiene las variables “Ciclo” y “IDCurso” es posible crear una consulta sobre el DF “Retiros” que modifique el DF “df” y que permita adicionar una nueva variable denominada “Retiros”, este nuevo dato hará parte del DF “df”.

```
[16] #Enriquecemos el DataSet ingresando los retiros por asignatura y periodo
     for i, row in df.iterrows():
         consulta = Retiros[(Retiros["ID ASIGNATURA"] == row["IDCurso"]) & (Retiros["Periodo"] == row["Ciclo"])]
         df.at[i,"Retiros"]=len(consulta)
```

Figura 12. Consulta al DF ‘Retiros’

Para completar la consulta es necesario iterar en el DF “df”, en cada paso de la iteración se realiza una consulta al DF “Retiros”, esta consulta válida si una asignatura tuvo retiros para un periodo específico, de ser así se valida el número de retiros y se adiciona esta cantidad a cada una de las asignaturas que conforman el DF “df”.

Con el ánimo de revisar que los datos relacionados con los retiros se agregaran de forma correcta, se realiza una consulta al DF denominado “df”, allí se selecciona una asignatura específica y se compara el resultado con el DF “Retiros”, el resultado de la comparación es el siguiente:

```
consulta=df[(df["Ciclo"] == 1610) & (df["IDCurso"] == 4196)]
consulta
```

nes	Martes	Miercoles	Jueves	Viernes	Sabado	Domingo	Acceso	FechaNovedad	EstadoAsociación	TotalInscripciones	NroModeloClase	RolProfesor	Retiros	Dias
N	N	N	N	Y	N	N	Contabilizar	NaT	Activo	24	2	Profesor	1.0	131
N	N	Y	N	N	N	N	Contabilizar	NaT	Activo	24	1	Profesor	1.0	131
N	N	N	Y	N	N	N	Contabilizar	NaT	Activo	18	2	Profesor	1.0	131
N	Y	N	N	N	N	N	Contabilizar	NaT	Activo	18	1	Profesor	1.0	131

Figura 13. Campo 'Retiros'

Se toma como variables de validación a la asignatura con ID '4196' del periodo '1610', se observa que hay un retiro el cual se debe contrastar contra el DF 'Retiros' para verificar su valides en la siguiente imagen se observa el resultado de la consulta:

```
consulta = Retiros[(Retiros["ID ASIGNATURA"] == 4196) & (Retiros["Periodo"] == 1610)]
consulta
#len(consulta)
```

	Cosecutivo	BECARIO	FECHA DE SOLICITUD	ID ASIGNATURA	ASIGNATURA	NUMERO DE CLASE	MOTIVO	Periodo	ESTADO ACADEMICO	Créditos	BECA	Unnamed: 9	Unnamed: 10
26	RA-20161-025	NO	2016-02-25	4196	Estructuras de Datos	1436	Bajo Rendimiento	1610	NaN	NaN	NaN	NaN	NaN

Figura 14. DF 'Retiros' Id 4196

El resultado evidencia que uno de los estudiantes inscritos se retiró de esta asignatura en el periodo 1610, esta información es coherente con el dato 'Retiros' que se encuentra indicado previamente.

Siguiendo un procedimiento similar al anterior, ahora se adiciona información relacionada con las calificaciones de las asignaturas vistas por los estudiantes para una asignatura, clase y periodo específico, por tanto, se crea un DF para cada periodo con base en los documentos de Excel entregados, el resultado es el siguiente:

```
[16] #df.describe()
     Notas.dtypes

Grado Académico            object
Programa Académico Base    object
Ciclo                       int64
ID Curso                    int64
Descripción                object
Nº Clase                    int64
Uni Matrd                   int64
Calif                     float64
Estado                     object
Nº Oferta                   int64
Sesión                     object
Sección                    object
Org Acad                   object
Estado.1                   object
Motivo                     object
dtype: object
```

Figura 15. Estructura del DF Notas

```
[18] #Enriquecemos el DataSet ingresando los estudiantes que perdieron o pasaron por asignatura, periodo y # de Clase
     for i, row in df.iterrows():
         consulta = Notas[(Notas["ID Curso"] == row["IDCurso"]) & (Notas["Ciclo"] == row["Ciclo"]) & (Notas["Nº Clase"] == row["Clase"])]
         PA=0
         PE=0
         if len(consulta) > 0:
           for j, row1 in consulta.iterrows():
             if row1["Calif"] >= 3.0 :
               PA += 1
             else:
               PE += 1
           df.at[i,"Perdidas"]=PE
           df.at[i,"Aprobados"]=PA
         else:
           df.at[i,"Perdidas"]=0
           df.at[i,"Aprobados"]=row["TotalInscripciones"]
```

Figura 16. Código en Python para adicionar las notas

Del anterior procedimiento es importante entender que si una calificación es inferior a 3.0 se toma como una clase perdida, si por el contrario el resultado supera esta cifra se da por hecho que la nota para esta clase es aprobatoria.

Continuando con el tratamiento de los datos es necesario dar formato a algunas de las variables por lo tanto se formatean los campos (NroHoras, NroSemanas, HorasSemanales) y se configuran como datos enteros.

```
#Formateamos las siguientes variables
df['NroHoras'] = df['NroHoras'].astype(int)
df['NroSemanas'] = df['NroSemanas'].astype(int)
df['HorasSemanales'] = df['HorasSemanales'].astype(int)
```

Figura 17. Formato de Variables

Posteriormente adicionamos otras variables, calculamos el número de días presentes entre la fecha de inicio de la clase y la fecha fin de la clase a este dato lo denominamos “Días”, el procedimiento para realizar el cálculo se describe en la siguiente imagen:

```
#Calculamos el número de días entre la fecha inicial y la fecha final de las clases
dias=(df['FechaFinal']-df['FechaInicial']).dt.days
df["Dias"]=dias
df['Dias'] = df['Dias'].astype(int)
```

Figura 18. Cálculo de la variable días

Debido a que el dataset “df” tiene registros duplicados por asignatura y periodo es necesario eliminar algunas columnas antes de usar este conjunto de datos, a continuación, se relaciona la instrucción de python usada para este fin, las columnas que se eliminaron fueron las siguientes:

“*index,Descr,IdInstal,FechaInicial,FechaFinal,OrgAcad,GpAcad,DocID,Nombre,NroContrato,EstadoContrato,FechaInicialAsociación,FechaFinalAsociación,Acceso,FechaNovedad,EstadoAsociación,RolProfesor,Catalogo,Grado,Seccion,EstadoClase,Modalidad,ID,HoraInicioAsociacion,HoraFinAsociacion,Lunes,Martes,Miercoles,Jueves,Viernes,Sabado,Domingo,NroModeloClase,IdModalidad, NroSemanas*”.

```
[26] #Eliminamos del dataframe algunas Columnas
     df=df.drop(['index','Descr','IdInstal','FechaInicial','FechaFinal','OrgAcad','GpAcad','DocID','Nombre','NroContrato','EstadoContrato',
                 'FechaInicialAsociación','FechaFinalAsociación','Acceso','FechaNovedad','EstadoAsociación',
                 'RolProfesor','Catalogo','Grado','Seccion','EstadoClase','Modalidad','ID','HoraInicioAsociacion','HoraFinAsociacion','Lunes','Martes','Miercoles','Jue
                 ,'Domingo','NroModeloClase','IdModalidad'],1)
```

Figura 19. Cálculo de la variable días

Se revisan el conjunto de datos vigente en el DF "df" después de eliminar las columnas indicadas previamente, es importante destacar que en este momento persisten los registros duplicados los cuales se eliminan posteriormente.

```
df.info() # Chequeemos los resultados

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3100 entries, 0 to 3099
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Sesion              3100 non-null   object
 1   IDCurso             3100 non-null   object
 2   Clase               3100 non-null   int64
 3   Ciclo               3100 non-null   object
 4   TipoClase           3100 non-null   object
 5   Componente          3100 non-null   object
 6   NroHoras            3100 non-null   int64
 7   NroSemanas          3100 non-null   int64
 8   HorasSemanales      3100 non-null   int64
 9   TotalInscripciones  3100 non-null   int64
 10  Retiros             3100 non-null   float64
 11  Dias                3100 non-null   int64
dtypes: float64(1), int64(6), object(5)
memory usage: 290.8+ KB
```

Figura 20. Variables DF "df"

```
[30] df1.head(5)
```

	Sesion	IDCurso	Clase	Ciclo	TipoClase	Componente	NroHoras	NroSemanas	HorasSemanales	TotalInscripciones	Retiros	Dias
0	1	4181	5721	2130	Sección Inscripción	Teorico Práctico	2	18	36	19	0.0	131
1	1	4056	2549	2130	Sección Inscripción	Teorico Práctico	2	18	36	19	0.0	131
2	1	4056	2549	2130	Sección Inscripción	Teorico Práctico	2	18	36	19	0.0	131
3	1	4056	2550	2130	Sección Inscripción	Teorico Práctico	2	18	36	20	0.0	131
4	1	4056	2550	2130	Sección Inscripción	Teorico Práctico	2	18	36	20	0.0	131

Figura 21. DF "df"

Para eliminar los registros duplicados agrupamos las siguientes variables (Sesion, IDCurso, Clase, Ciclo, TipoClase, Componente, TotalInscripciones, Retiros, Dias) y sumamos estas (NroHoras, HorasSemanales), la instrucción usada para realizar este procedimiento es la siguiente:

```
[ ] df1=df1.groupby(['Sesion','IDCurso','Clase','Ciclo','TipoClase','Componente','NroSemanas','TotalInscripciones'
    ,'Retiros','Dias'])[['NroHoras','HorasSemanales']].sum()
```

Figura 22. Agrupación de las variables DF "df"

Después de agrupar el DF "df" es necesario reiniciar los índices para continuar operando con normalidad, el siguiente paso es eliminar la variable 'Clase' y agrupar nuevamente por los campos: "Sesion, IDCurso, Ciclo, TipoClase, Componente, NroSemanas, Retiros, Dias, NroHoras", esto se hace con el ánimo de sumar la variable "TotalInscripciones" para hallar el número de estudiantes inscritos por asignatura para un período específico, nuevamente es necesario reiniciar los índices.

```
[44] df1.head(10)
```

	Sesion	IDCurso	Clase	Ciclo	TipoClase	Componente	TotalInscripciones	Retiros	Perdidas	Aprobados	Dias	NroHoras	HorasSemanales
0	1.0	3194.0	1005.0	1630.0	Sección Inscripción	Teorico Práctico	27.0	0.0	6.0	21.0	131.0	4.0	72.0
1	1.0	3194.0	1005.0	1710.0	Sección Inscripción	Teorico Práctico	23.0	0.0	0.0	23.0	131.0	4.0	72.0
2	1.0	3194.0	1006.0	1510.0	Sección Inscripción	Teorico Práctico	20.0	0.0	0.0	20.0	131.0	3.0	54.0

Figura 23. Reinicio Índices DF "df"

```
[45] df1= df1.drop_duplicates()

[46] df1=df1.drop(['Clase'],1)

[47] df1=df1.groupby(['Sesion','IDCurso','Ciclo','TipoClase','Componente',
                      'Dias','NroHoras','HorasSemanales'])[['TotalInscripciones','Retiros','Perdidas','Aprobados']].sum()

[48] df1=pd.DataFrame(df1)

[49] df1=df1.reset_index()

[50] df1.head(20)
```

Figura 24. Agrupación II de las variables DF "df"

Debido a que se adicionaron datos relacionados con las calificaciones y retiros de clases por parte de los estudiantes, es posible calcular la probabilidad de pasar una asignatura, el cálculo se realiza dividiendo el número de aprobados entre el número de inscritos para un periodo específico (ProbAprobar = Número de aprobados /Número de inscritos), se itera en el DF "df"para adicionar esta cifra a cada una de las líneas que contienen la información de un semestre y asignatura. Adicionalmente se agrega una nueva columna con el número de inscritos por asignatura para el siguiente periodo, esto en función de crear la variable supervisada "Y".

```
[55] #Creamos la Variable #InscritosSiguiente Periodo
     df["InscritosSiguientePeriodo"] = 0
     dfPeriodo = pd.DataFrame()
     dfPeriodo['Periodo'] = [1510, 1530, 1610, 1630, 1710, 1730, 1810, 1830, 1910, 1930, 2010, 2030, 2110, 2130]

     for i, row in df.iterrows():
       consulta = dfPeriodo.index[(dfPeriodo["Periodo"] == row['Ciclo'])]
       for data in consulta:
         data += 1
         data1 = data + 1
         Periodo=dfPeriodo.iloc[data:data1, :]
         #print(Periodo["Periodo"])
         for k, row3 in Periodo.iterrows():
           Ciclo=row3["Periodo"].astype(int)
           #print(Ciclo)

         consulta = df[(df["IDCurso"] == row["IDCurso"]) & (df["Ciclo"] == Ciclo)]
         if len(consulta) > 0:
           for j, row1 in consulta.iterrows():
             df.at[i,"InscritosSiguientePeriodo"]=row1["TotalInscripciones"] + row1["Retiros"]
         else:
           df.at[i,"InscritosSiguientePeriodo"]=0
```

Figura 25. Creación de la Variable Inscritos Siguiente Periodo

Cabe resaltar que para este punto del desarrollo se involucran variables nuevas con base en el diseño de un *Grafo dirigido*, se dispone del uso de la librería *NetworkX de Python* para modelar el plan de estudios del programa de Ing. de Sistemas, se adicionan nodos, los cuales corresponden a los Id de las asignaturas que lo componen y enlaces (Edge) los cuales corresponden a las dependencias entre asignaturas, es importante destacar que estas dependencias se encuentran estructuradas con base plan de estudios del programa. En la siguiente imagen se observan las líneas de código usadas para crear los enlaces entre los diferentes nodos.

```
[4]  #cargamos los datos de entrada
     df = pd.read_excel(r"/content/drive/MyDrive/DiagramaGrafos.xlsx")
```

```
[5]  df.head()
```

	Id	Asignatura	IdEdge	AsignaturaEdge
0	1295	Cálculo Diferencial	1297	Cálculo Integral
1	1297	Cálculo Integral	33732	Probabilidad y Estadística
2	1297	Cálculo Integral	1300	Ecuaciones Diferenciales
3	1297	Cálculo Integral	1299	Cálculo Vectorial
4	33732	Probabilidad y Estadística	34866	Opti. Y Simulación

Figura 26. Nodos Grafo NetworkX - Plan de Estudios

La variable Id corresponde al número de Id de asignaturas precedentes de otras asignaturas denominadas IdEdge. A continuación, se observa el código en Python que permite implementar el Grafo correspondiente.

```
[9]  for indice, i in df.iterrows():
         graph.add_edge(i['Id'], i['IdEdge'])
```

```
[10] print("The network of RT of the hashtag was analyzed succesfully.")
     print("Saving the file as Grafo #1")
     posicion = nx.circular_layout(graph)
     nx.draw(graph, pos=posicion, with_labels = True)
```

```
The network of RT of the hashtag was analyzed succesfully.
Saving the file as Grafo #1
```

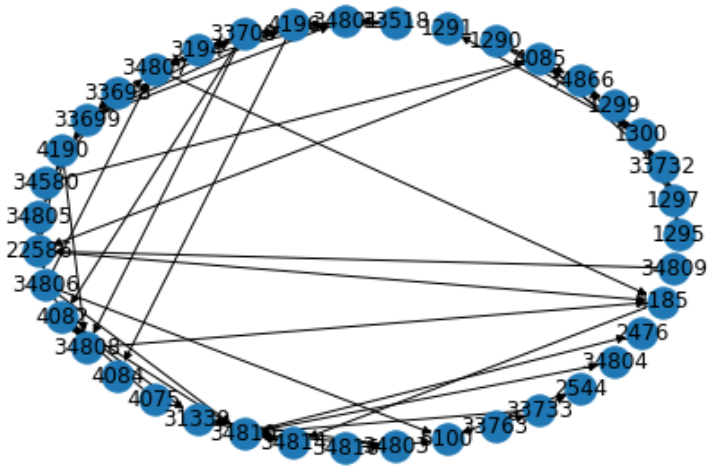


Figura 27. Gráfico del Grafo NetworkX - Plan de Estudios

```
[13] all_degrees = graph.degree()
     df_net_all = pd.DataFrame(all_degrees, columns =['node', 'degree'])
     df_net_all_group = df_net_all.groupby(by=['degree']).count().reset_index()
     plt.figure()
     plt.plot(df_net_all_group['degree'], df_net_all_group['node'],'gx-')
     plt.legend(['In-degree','Out-degree', 'All-degree'])
     plt.xlabel('Degree')
     plt.ylabel('Number of nodes')
     plt.title('Rand - Degree')
     plt.savefig('Rand-degree.pdf')
     plt.show()
     plt.close()
```

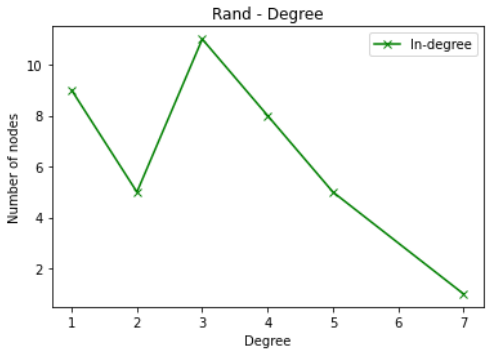


Figura 28. Gráfico del Grafo NetworkX - Plan de Estudios

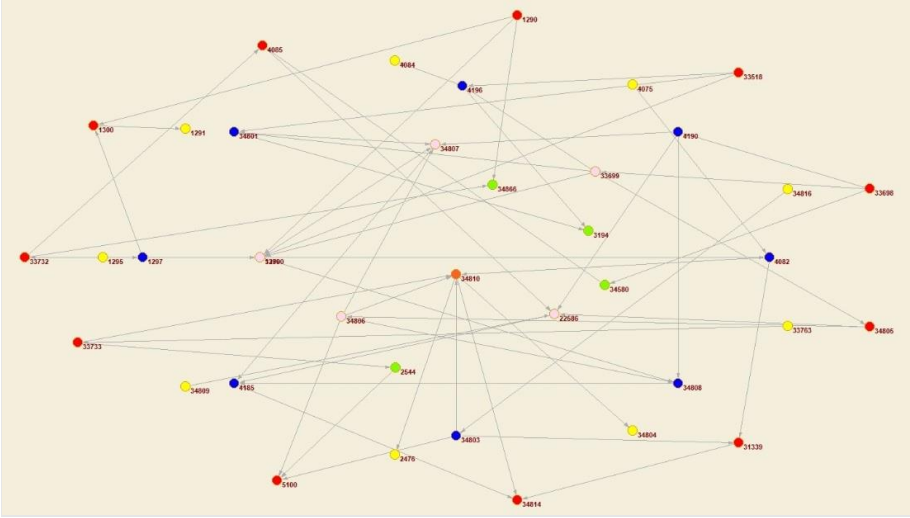


Figura 29. Grafo NetworkX - Plan de Estudios - All Degree

Al ser un grafo dirigido es posible modelar el comportamiento de las asignaturas que preceden a otras signaturas, por tanto, es posible revisar características propias de esta estructura, como, por ejemplo:

- Grado de un nodo
- Centralidad de un nodo
- Suma de los estudiantes que se encuentran inscritos en asignaturas precedentes de un nodo específico.

En la anterior imagen es posible observar la descripción del grado (All degree) de cada uno de los nodos que componen el plan de estudios, entendiendo que un nodo corresponde al Id de una asignatura, en donde hallamos que la media de los grados del *Grafo* es 3 y hay pocas asignaturas que tienen un grado de 7, sería válido entender que estas últimas formarían un cuello de botella por lo tanto se considerarían nodos más importantes que los otros.

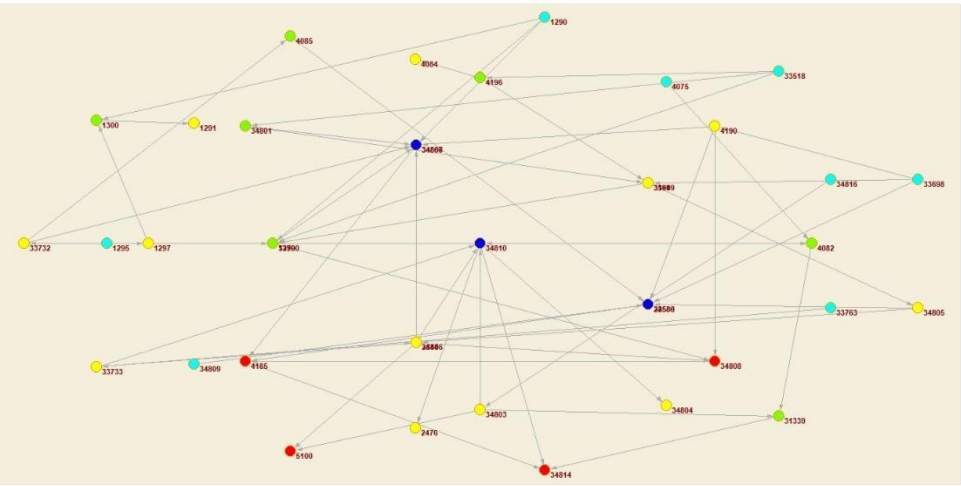


Figura 30. Grafo NetworkX - Plan de Estudios – Input Degree

La estructura Input Degree se divide en varios grupos, los nodos que se encuentran marcados de color azul tienen un grado de entrada de 4, los marcados en color amarillo tienen un grado de entrada de 1, los marcados en color azul celeste tienen un grado de entrada de 0, los marcados en color verde tienen un grado de entrada de 2, los marcados en color rojo tienen un grado de entrada de 3. Mayoritariamente los nodos predominantes en esta estructura son los que tiene un grado de entrada de 1, los cuales corresponden al grupo amarillo de los que se cuentan 14 Id, seguidos de los de color Azul Celeste de los cuales hay 11 Id. De la anterior observación se entiende que hay varias asignaturas que no tiene prerrequisitos, pero mayoritariamente tiene por lo menos uno.

Con base en las anteriores observaciones y con el ánimo de enriquecer el conjunto de datos de entrenamiento se dispuso de la implementación de algunas líneas de Python que tiene por fin adicionar 3 columnas nuevas al conjunto de datos:

- Sum_antecestors: Este campo corresponde a la suma de los estudiantes inscritos en las asignaturas precedentes de una asignatura que tenga por lo menos una asignatura como requisito de inscripción.
- Grados: Representa el Grado (All Degree) de un nodo especifico.
- Centralidad: Representa la importancia de un nodo específico respecto de los demás nodos que componen el grafo.

```
[ ]  #Enrriquecemos el Dataset Incluyendo información del Grafo del Nuevo plan de Estudios
     for i, row in df.iterrows():
       if row["IDCurso"] in nodos:
         Grados=nx.degree(graph, np.int64(row["IDCurso"]))
         ancestors=nx.ancestors(graph, np.int64(row["IDCurso"]))
         sum_aprobados=0
         for a in ancestors:
           consulta=df[(df["Ciclo"] == row["Ciclo"]) & (df["IDCurso"] == a)]
           for b, data in consulta.iterrows():
             aprobados=data["TotalInscripciones"]
             if aprobados > 0:
               sum_aprobados = sum_aprobados + aprobados
         Centrality=nx.degree_centrality(graph)
         df.at[i,"sum_antecestors"]=sum_aprobados
         df.at[i,"Grados"]=Grados
         df.at[i,"Centralidad"]=Centrality[row["IDCurso"]]
       else:
         df.at[i,"Grados"]=0
         df.at[i,"sum_antecestors"]=0
         df.at[i,"Centralidad"]=0
```

Figura 31. Código en Python para consultar el grafo del plan de estudios del programa de Ingeniería de Sistemas

Para ejecutar el conjunto de pruebas disponemos de un nuevo set de variables, las cuales se describen a continuación:

- Sesión: Corresponde al calendario usado para programar una asignatura, es decir el número de semanas que la componen.
- IDCurso: Corresponde al Id de la asignatura.
- Ciclo: Corresponde al periodo académico en el cual se programó la asignatura.
- TipoClase: Sección de inscripción o Sección sin inscripción.
- Días: Días transcurridos desde la fecha inicial de la clase hasta la fecha final de la asignatura.
- NroHoras: Número de horas por cada modelo de reunión, es decir el número de horas que la asignatura se dicta a la semana.
- HorasSemanales: Corresponde al número total de horas de clase de una asignatura al semestre.
- TotalInscripciones: Corresponde al número de estudiantes inscritos para una asignatura en un periodo especifico.
- Retiros: Corresponde al número de estudiantes que se retiran de la asignatura para un periodo específico.
- Perdidas: Corresponde al número de estudiantes que pierden la asignatura para un periodo específico.
- Aprobados: Corresponde al número de estudiantes que aprueban la asignatura para un periodo específico.
- ProbAprobar: Corresponde a la probabilidad de que un estudiante aprueba la asignatura para un periodo específico.
- InscritosSiguientePeriodo: (*Variable Objetivo)*, Corresponde al número de estudiantes que se inscribieron en la asignatura para el siguiente periodo académico.
- Grados: (*All Degree)*, número total de grados de un nodo.
- sum_antecestors: Este campo corresponde a la suma de los estudiantes inscritos en las asignaturas precedentes de una asignatura que tenga por lo menos una asignatura como requisito de inscripción.

- Centralidad: Representa la importancia de un nodo específico respecto de los demás nodos que componen el grafo.

La siguiente instrucción de Python se usa para revisar la variedad y el número de datos de los campos del tipo ‘category’, ‘object’ y ‘bool’, se encuentra que la mayoría de las clases se registran como ‘Sección Inscripción’ del componente ‘Teórico Práctico’, seguido del componente ‘Teorico’.

```
# El siguiente es el código para chequear todas las columnas:
for col in df.select_dtypes(include=['category','object','bool']).columns:
    print(col)
    print(df[col].value_counts())
    print('\n')
```

```
TipoClase
Sección Inscripción        751
Sección sin Inscripción     17
Name: TipoClase, dtype: int64


Componente
Teorico Práctico    497
Teorico             140
Práctico             48
Laboratorio          28
Proyecto             28
Trabajo de Campo     14
Salida de Campo       8
Seminario             5
Name: Componente, dtype: int64
```

Figura 32. Columnas DF ‘df’

Al igual que con el DF “Retiros” se revisa si el DF “df” tiene datos faltantes, se observa que no hay datos perdidos, pero de igual forma se desarrolla una función que tiene por objetivo identificar si hay datos de este tipo.

```
[63] # Limpieza de datos -- parte 3: Reemplazo/Imputación de datos faltantes

     pd.DataFrame(df).isna().sum() # Chequeamos si hay datos faltantes
```

```
Sesion                      0
IDCurso                     0
Ciclo                       0
TipoClase                   0
Componente                  0
Dias                        0
NroHoras                    0
HorasSemanales              0
TotalInscripciones          0
Retiros                     0
Perdidas                    0
Aprobados                   0
ProbAprobar                 0
InscritosSiguientePeriodo   0
Grados                      0
sum_antecestors             0
Centralidad                 0
dtype: int64
```

Figura 33. Datos Faltantes DF ‘df’’

```
[64] pd.set_option('display.max_rows', 500)
     # Hay datos no disponibles.  Revisemos el porcentaje de los mismos.
     nan_percentage = pd.DataFrame(df).isna().sum() / len(pd.DataFrame(df))
     missing_percentage_df = pd.DataFrame({'column_name': pd.DataFrame(df).columns, 'percent_missing': nan_percentage}).reset_index(drop=True)
     missing_percentage_df
```

	column_name	percent_missing
0	Sesion	0.0
1	IDCurso	0.0
2	Ciclo	0.0
3	TipoClase	0.0
4	Componente	0.0

Figura 34. Función Datos Faltantes DF 'df'

Para completar los datos faltantes en caso de que existan, se dispone del desarrollo de una función que implemente los métodos de imputación disponibles en la librería "Skilearn", esta función se compone de dos partes, la primera tiene por objetivo identificar los tipos de datos 'Categóricos' para usar la estrategia Constante "strategy='constant'" y reemplazar los datos perdidos con un texto que los identifique, el cual se denomina "Constant". La segunda parte consiste en identificar los tipos de dato numéricos para usar la estrategia Mediana "strategy='median'" y reemplazar los datos perdidos con la media de los demás datos del mismo campo. En la siguiente imagen se observan las líneas de código usadas.

```
# creación de variables sustitutas
for col in df:
    if df[col].isna().sum() != 0:
        df[col + '_surrogate'] = df[col].isna().astype(int)

# fijación de variables categóricas
imputer = SimpleImputer(missing_values = np.nan, strategy='constant')
imputer.fit(df.select_dtypes(exclude=['int64','float64']))
df[df.select_dtypes(exclude=['int64','float64']).columns] = imputer.transform(df.select_dtypes(exclude=['int64','float64']))

# fijación de variables numéricas
imputer = SimpleImputer(missing_values = np.nan, strategy='median')
imputer.fit(df.select_dtypes(include=['int64','float64']))
df[df.select_dtypes(include=['int64','float64']).columns] = imputer.transform(df.select_dtypes(include=['int64','float64']))

# Examinemos los resultados para la variable "Poverty.Code"
#df[['IdModalidad','IdModalidad_surrogate']]
```

Figura 35. Función Imputación Datos Faltantes DF 'df'

Revisamos los tipos de datos de los campos del DF y los ajustamos los categóricos y los numéricos, este paso es necesario antes de completar las transformaciones posteriores.

```
[69] df['Sesion'] = df['Sesion'].astype('object')
     df['IDCurso'] = df['IDCurso'].astype('object')
     df['Ciclo'] = df['Ciclo'].astype('object')
     df['TipoClase'] = df['TipoClase'].astype('object')
     df['Componente'] = df['Componente'].astype('object')
     df['Retiros']=df['Retiros'].astype(int)
     df['Dias']=df['Dias'].astype(int)
     df['NroHoras']=df['NroHoras'].astype(int)
     df['HorasSemanales']=df['HorasSemanales'].astype(int)
     df['TotalInscripciones']=df['TotalInscripciones'].astype(int)
```

Figura 36. Tipos de datos DF 'df'

```
[72] #Creamos un nuevo Dataset de Pruebas para imprimirlo al final
     dfpredicciones=df.query('Ciclo == 1810')
     dfpredicciones.head(10)
```

	Sesion	IDCurso	Ciclo	TipoClase	Componente	Dias	NroHoras	HorasSemanales	TotalInscripciones	Retiros	Perdidas	Aprobados	ProbAprobar	InscritosSiguientePeriodo
6	1	3194	1810	Sección Inscripción	Teorico Práctico	131	4	72	40	2	2.0	38.0	0.950000	43.0
15	1	4055	1810	Sección Inscripción	Teorico Práctico	131	3	54	20	0	0.0	18.0	0.900000	0.0
25	1	4056	1810	Sección Inscripción	Teorico Práctico	131	4	72	115	0	0.0	115.0	1.000000	103.0
48	1	4064	1810	Sección Inscripción	Teorico	131	4	72	28	0	0.0	28.0	1.000000	35.0
68	1	4067	1810	Sección Inscripción	Laboratorio	131	3	54	219	0	0.0	219.0	1.000000	204.0

Figura 37. DF "df"

Posteriormente se crea una copia del dataset "df" filtrado por el periodo académico usado para realizar las pruebas según el esquema que corresponda, en la imagen previa se filtró el dataset por el periodo "1810". Este nuevo conjunto de datos se denominó "dfpredicciones" y se concatenará al final con los resultados del modelo evaluado.

Debido a que los modelos que se van a implementar en python únicamente soportan tipos de datos numéricos, es necesario disponer del algoritmo 'OneHotEncoding' para tratar los datos categóricos que para este momento son los siguientes:

- Sesión
- IdCurso
- Ciclo
- TipoClase
- Componente

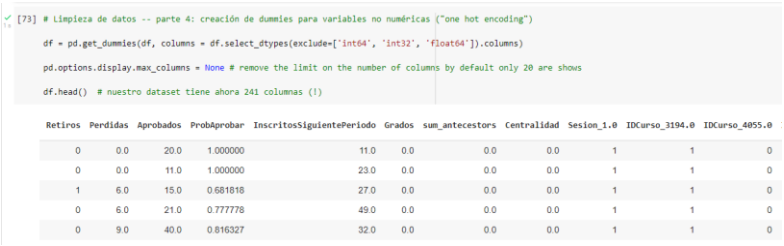
Los demás datos son del tipo (Int64):

- Retiros
- Dias
- NroHoras
- HorasSemanales
- TotalInscripciones
- Retiros
- Perdidas
- Aprobados
- ProbAprobar
- InscritosSiguientePeriodo
- Grados
- sum_antecestors
- Centralidad

```
[50] # Limpieza de datos -- parte 4: creación de dummies para variables no numéricas ("one hot encoding")

     df = pd.get_dummies(df, columns = df.select_dtypes(exclude=['int64', 'int32', 'float64']).columns)

     pd.options.display.max_columns = None # remove the limit on the number of columns by default only 20 are shows

     df.head()  # nuestro dataset tiene ahora 241 columnas (!)
```

Figura 38. One Hot Encoding DF 'df'

Como se observa en la siguiente imagen el algoritmo “One Hot Encoding” creo nuevas columnas para los datos categóricos y clasifico como ‘1’ o ‘0’ a los nuevos campos, es decir por ejemplo para el campo ‘IdCurso’ creo tantos nuevos campos como números de Id existentes en el dataset, la clasificación es del orden binario en donde, 1 representa que ese registro pertenece a una asignatura especifica, sí por el contrario aparece un 0 índica que esta asignatura no pertenece a este registro. De igual forma para cada uno de los datos categóricos presentes en el DF ‘df’.



```
[73] # Limpieza de datos -- parte 4: creación de dummies para variables no numéricas ("one hot encoding")

     df = pd.get_dummies(df, columns = df.select_dtypes(exclude=['int64', 'int32', 'float64']).columns)

     pd.options.display.max_columns = None # remove the limit on the number of columns by default only 20 are shows

     df.head()  # nuestro dataset tiene ahora 241 columnas (!)
```

Retiros	Perdidas	Aprobados	ProbAprobar	InscritosSiguientePeriodo	Grados	sum_antecestors	Centralidad	Sesion_1.0	IDCurso_3194.0	IDCurso_4055.0
0	0.0	20.0	1.000000	11.0	0.0	0.0	0.0	1	1	0
0	0.0	11.0	1.000000	23.0	0.0	0.0	0.0	1	1	0
1	6.0	15.0	0.681818	27.0	0.0	0.0	0.0	1	1	0
0	6.0	21.0	0.777778	49.0	0.0	0.0	0.0	1	1	0
0	9.0	40.0	0.816327	32.0	0.0	0.0	0.0	1	1	0

Figura 39. One Hot Encoding DF ‘df’ II

4. Modelado

4.1.Selección de la técnica de Modelado

Support Vector Machine: Es una técnica clásica de aprendizaje de máquina principalmente diseñada para resolver problemas de clasificación, pero adaptada también para responder de forma efectiva en problemas de regresión, sin embargo, es matemáticamente compleja y de alto costo computacional.

Existen varios métodos para implementar este tipo de modelos, aparte de ajustar los hiperparametros, es necesario revisar cuál de los núcleos disponibles presenta mejor desempeño. Durante el desarrollo del modelo se implementaron 3 tipos de Núcleo (Linear, Radial y Polinomial), que básicamente consiste en transformar las observaciones originales a un espacio de diferente dimensionalidad en el que pueda dibujarse un hiperplano.

Redes Neuronales: Es un modelo matemático computacional basado en redes neuronales biológicas, que consiste en interconectar un grupo de neuronas artificiales y procesarlas. Para este caso de estudio específico su diseño centra en la solución de un problema de regresión.

4.2.Generar Pruebas de Diseño

Durante el proceso iterativo de desarrollo de los primeros modelos se dividió el conjunto de entrenamiento y pruebas así:

- Entrenamiento: (1510, 1530, 1610, 1630, 1710, 1730, 1810, 1830, 1910, 1930, 2010, 203 y 2110).
- Pruebas: (2130)

Posteriormente se fija un esquema de experimentación nuevo para todas las pruebas de aquí en adelante, en donde se toman 3 periodos académicos para entrenar los diferentes modelos y un periodo académico para probarlos.

A continuación, se relaciona el esquema usado para evaluar el diseño de los modelos, los datasets de entrenamiento y validación se dividieron así:

- Primer conjunto de datos de entrenamiento (1510, 15030 y 1610)
- Primer conjunto de Pruebas (1630)

- Segundo conjunto de datos de entrenamiento (1630, 1710 y 1730)
- Segundo conjunto de datos de Pruebas: (1810)

- ★ Tercer conjunto de datos de entrenamiento (1810, 1830 y 1910)
- ★ Tercer conjunto de datos de Pruebas (1930)

- ★ Cuarto conjunto de datos de entrenamiento (1930, 2010, 2030)
- ★ Cuarto conjunto de datos de pruebas (2130)

4.3.Construcción del Modelo

En el desarrollo del proyecto se consideraron en primer lugar las necesidades del negocio y como la tecnología se podía alinear con ellas, es decir se buscó integrar en el desarrollo un modelo que respondiera de forma efectiva a cada uno de los objetivos planteados, debido a que este proyecto involucra como componente principal la supervisión de una variable Y (Número total de inscritos para una asignatura en un periodo específico) con base en características X, se consideró realizar un tratamiento previo de las variables y una selección posterior de las que mejor explican el comportamiento de la variable objetivo, es necesario señalar que para el primer conjunto de ensayos que se revisara a continuación se usaron todas las variables disponibles, posteriormente se eliminan algunas de ellas y se crean otras. El conjunto de datos se dividió en dos: entrenamiento y pruebas, inicialmente se dispuso del 80% de los datos para entrenar el modelo y el 20% restante para pruebas; pero se decidió caracterizar los datos en función de su uso, es decir agrupar los datos por periodo académico, desde el 1510 hasta el 2030 para entrenar el modelo, y 2110 para realizar pruebas. Es necesario señalar que esta es solamente una de las pruebas realizadas para encontrar el mejor modelo, más delante se realizan nuevos experimentos y se dispone del uso de validación cruzada para mejorar la precisión.

```
In [460]: df1510 = df[(df["Ciclo_1510.0"] == 1)]
          df1530 = df[(df["Ciclo_1530.0"] == 1)]
          df1610 = df[(df["Ciclo_1610.0"] == 1)]
          df1630 = df[(df["Ciclo_1630.0"] == 1)]
          df1710 = df[(df["Ciclo_1710.0"] == 1)]
          df1730 = df[(df["Ciclo_1730.0"] == 1)]
          df1810 = df[(df["Ciclo_1810.0"] == 1)]
          df1830 = df[(df["Ciclo_1830.0"] == 1)]
          df1910 = df[(df["Ciclo_1910.0"] == 1)]
          df1930 = df[(df["Ciclo_1930.0"] == 1)]
          df2010 = df[(df["Ciclo_2010.0"] == 1)]
          df2030 = df[(df["Ciclo_2030.0"] == 1)]

In [461]: #Dividimos el Dataset para Pruebas del periodo 1510 - 2030
          df_entrenamiento=pd.concat([df1510, df1530])
          df_entrenamiento=pd.concat([df_entrenamiento, df1610])
          df_entrenamiento=pd.concat([df_entrenamiento, df1630])
          df_entrenamiento=pd.concat([df_entrenamiento, df1710])
          df_entrenamiento=pd.concat([df_entrenamiento, df1730])
          df_entrenamiento=pd.concat([df_entrenamiento, df1810])
          df_entrenamiento=pd.concat([df_entrenamiento, df1830])
          df_entrenamiento=pd.concat([df_entrenamiento, df1910])
          df_entrenamiento=pd.concat([df_entrenamiento, df1930])
          df_entrenamiento=pd.concat([df_entrenamiento, df2010])
          df_entrenamiento=pd.concat([df_entrenamiento, df2030])
```

Figura 40. División del conjunto de datos de Entrenamiento y Pruebas

Con el fin de establecer cuál era el modelo que presentaba mejor desempeño se decidió usar como métrica el error cuadrático medio RMSE, el cual consiste en obtener la raíz cuadrada de la suma de las diferencias de los errores del conjunto de predicciones vs el conjunto real de datos.

Modelo 1: Se diseña un modelo SVM con núcleo radical sobre el conjunto de datos de entrenamiento.

```
In [472]: from sklearn.svm import SVR
          regressor = SVR(kernel = 'rbf')
          regressor.fit(X_train, y_train)

Out[472]: SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
              gamma='auto_deprecated', kernel='rbf', max_iter=-1, shrinking=True,
              tol=0.001, verbose=False)
```

Figura 41. Código en Python algoritmo SVM Kernel Radial

Posteriormente se evalúa el desempeño del modelo en el conjunto de datos de prueba, como se indicó previamente el conjunto de datos usado para pruebas es el periodo 2110.

```
In [473]: y_pred = regressor.predict(X_test)
          y_pred = sc_y.inverse_transform(y_pred)
          y_pred

Out[473]: array([25.61008523, 22.56995404, 22.85329088, 23.6149691 , 21.30466854,
                 23.53216681, 21.12239542, 21.13500594, 21.85745206, 21.77526294,
                 21.40517098, 21.31720362, 24.1791208 , 24.01017556, 26.65584229,
                 26.68903393, 24.50113876, 24.27135136, 22.07889686, 22.12449442,
                  9.55129391,  9.4025745 , 27.0856297 , 18.68640196, 19.03420471,
                 22.27182761, 22.25626861, 23.42698396, 23.399328  , 22.91389707,
                 22.93723151, 22.93657239, 25.10745857, 25.33099392, 25.17420106,
                 23.99368112, 23.91426571, 23.96143337, 21.41388942, 21.40689546,
                 22.37941095, 22.40960165, 22.38447037, 22.93451499, 20.23594598,
                 24.93195843, 24.89339984, 24.82526121, 24.80325567, 23.62499989,
                 23.57329036, 22.87053444, 22.88052118, 23.57831146, 23.68024283,
                 20.48064697, 29.62282399, 28.78224683, 19.34191355, 19.55877848,
                 20.07477903, 19.87902477, 17.98524974, 17.88230747, 16.9316335 ,
                 16.8858704 , 29.68972012, 26.03047512, 24.74216114, 32.50585283,
                 30.88202153, 29.57983273, 30.11379646, 30.35432964, 25.6375265 ,
                 28.75133409, 23.48610557, 22.64188451, 24.02031366, 21.39599959,
                 20.73127868, 20.68626316, 20.91870896, 21.51208062, 21.86000741,
                 19.15104671, 21.5966415 , 20.59347966, 20.7709544 , 22.08694825,
                 23.24892249, 23.46275352, 24.89042301, 22.65961954, 22.94398245,
                 23.24096286, 21.7308278 , 21.45513245, 21.05463731, 20.90250491,
                 23.46695623, 23.0537884 , 23.45206449, 23.15600371, 21.67264293,
                 21.60822745, 21.64911917, 21.34665694, 20.73329044, 20.54759691,
                 19.44630871, 19.3312451 , 19.70463488, 19.77891197, 21.77724734,
                 22.0257422 , 16.15815023, 16.98201434, 17.95281631, 20.77396111,
                 25.5847397 , 27.65814126, 20.17822687, 17.53574446, 19.17122023,
```

Figura 42. Predicciones de la variable objetivo algoritmo SVM Kernel Radial - 2110

Creamos una tabla de validación de los datos reales vs. los datos predichos, esto lo hacemos con el ánimo de revisar el desempeño del modelo propuesto.

```
In [474]: df = pd.DataFrame({'RealValues':sc_y.inverse_transform(y_test.reshape(-1)), 'PredictedValues':y_pred})
          df
```

Out[474]:

	RealValues	PredictedValues
0	21.0	25.610085
1	18.0	22.569954
2	18.0	22.853291
3	19.0	23.614969
4	19.0	21.304669
5	19.0	23.532167
6	18.0	21.122395

Figura 43. Predicciones vs. valor real SVM Kernel Radial - 2110

Con estos datos evaluamos el rendimiento del modelo bajo la métrica RMSE (error cuadrático medio), es necesario señalar que entre más alto sea el resultado de esta medida el desempeño del modelo será más bajo, el resultado de la evaluación es 7.92.

```
In [475]: #importar las bibliotecas necesarias
          from sklearn.metrics import mean_squared_error
          from math import sqrt

          #calcular RMSE
          sqrt (mean_squared_error (df['RealValues'],df['PredictedValues']))

Out[475]: 7.9280243009964
```

Figura 44. RMSE del modelo SVM Kernel Radial – 2110

Modelo 2: Con base en el resultado del modelo planteado con un kernel radial, diseñamos uno diferente con un kernel "linear"

```
In [380]: #Probamos usando un kernel Linear
          from sklearn.svm import SVR
          regressor = SVR(kernel='linear')
          regressor.fit(X_train, y_train)

Out[380]: SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
              gamma='auto_deprecated', kernel='linear', max_iter=-1, shrinking=True,
              tol=0.001, verbose=False)
```

Figura 45. Código en Python algoritmo SVM Kernel Linear

De igual forma que en la instancia previa evaluamos el resultado del modelo en el conjunto de datos de pruebas, estas son las predicciones realizadas.

```
In [381]: y_pred = regressor.predict(X_test)
          y_pred = sc_y.inverse_transform(y_pred)
          y_pred

Out[381]: array([ 3.37328949e+01,  2.27995665e+01,  2.29969132e+01,  2.34657164e+01,
                  2.33123855e+01,  2.35868915e+01,  2.15810235e+01,  2.10853657e+01,
                  2.16443392e+01,  2.11486814e+01,  2.40165660e+01,  2.41961536e+01,
                  2.19980670e+01,  2.21776546e+01,  2.56294605e+01,  2.57771982e+01,
                  2.42353127e+01,  2.37396549e+01,  1.95377052e+01,  1.96854429e+01,
                  1.45872782e+01,  1.47350159e+01,  2.70497290e+01,  1.74123017e+01,
                  1.74782744e+01,  2.10658325e+01,  2.11318053e+01,  2.27015316e+01,
                  2.28492693e+01,  2.26959543e+01,  2.22499054e+01,  2.23158782e+01,
                  2.30170962e+01,  2.25710474e+01,  2.26370202e+01,  2.31333875e+01,
                  2.26873387e+01,  2.27533114e+01,  2.06499145e+01,  2.07976521e+01,
                  1.84504814e+01,  1.85982190e+01,  1.97267807e+01,  2.02177546e+01,
                  2.04105742e+01,  2.63842534e+01,  2.58885955e+01,  2.50579107e+01,
                  2.52056484e+01,  2.23260213e+01,  2.24737590e+01,  2.28980689e+01,
                  2.24024111e+01,  2.22132697e+01,  2.23610073e+01,  2.22064139e+01,
                  3.71167742e+01,  3.64701913e+01,  2.11093383e+01,  2.06136805e+01,
                  2.02270285e+01,  2.03747662e+01,  1.95843395e+01,  1.90886817e+01,
                  1.78491219e+01,  1.78459345e+01,  2.72417171e+01,  2.69337505e+01,
                  2.49617725e+01,  2.92820206e+01,  2.89843895e+01,  2.72779027e+01,
                  2.77677985e+01,  2.70556770e+01,  2.71777325e+01,  2.94317759e+01,
                  2.63560877e+01,  2.24393214e+01,  2.31699250e+01,  2.15325401e+01,
```

Figura 46. Predicciones de la variable objetivo algoritmo SVM Kernel Linear - 2110

Revisamos por medio del uso de una tabla de comparación los valores predichos vs. los valores reales en el conjunto de datos de prueba (2110), el proceso de ajuste del modelo sea hace de forma iterativa comparado los resultados obtenidos y el valor de las métricas de una prueba a otra.

```
In [382]: df = pd.DataFrame({'RealValues':sc_y.inverse_transform(y_test.reshape(-1)), 'PredictedValues':y_pred})
          df
```

Out[382]:

	RealValues	PredictedValues
0	21.0	33.732895
1	18.0	22.799567
2	18.0	22.996913
3	19.0	23.465716
4	19.0	23.312385
5	19.0	23.586892
6	18.0	21.581023
7	18.0	21.085366
8	18.0	21.644339
9	18.0	21.148681

Figura 47. Predicciones vs. valor real SVM Kernel Linear - 2110

Con estos datos evaluamos el rendimiento del modelo bajo la métrica RMSE (error cuadrático medio), es necesario señalar que entre más alto sea el resultado de esta medida el desempeño del modelo será más bajo, el resultado de la evaluación de este segundo modelo usando un kernel linear fue de 6.92 un poco más bajo que al usar un kernel Radial.

```
In [383]: #calcular RMSE
          sqrt (mean_squared_error (df['RealValues'],df['PredictedValues']))

Out[383]: 6.9215916433743585
```

Figura 48. RMSE del modelo SVM Kernel Linear – 2110

Modelo 3: Dentro del proceso de experimentación se dispone del desarrollo de un SVM usando un núcleo Polinomial que considere los siguientes hiperparametros, (C=5, gamma=0.05, degree=2), el resultado conseguido es el siguiente:

```
#Probamos usando un kernel Polynomial
from sklearn.svm import SVR
regressor = SVR(kernel='poly', C=5, gamma=0.05, degree=2)
regressor.fit(X_train, y_train)

SVR(C=5, cache_size=200, coef0=0.0, degree=2, epsilon=0.1, gamma=0.05,
    kernel='poly', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

Figura 49. Código en Python algoritmo SVM Kernel Poliomial- 2110

Se usa el modelo desarrollado previamente para predecir la variable objetivo sobre el conjunto de datos de prueba (2110), el resultado es el siguiente:

```
[66] y_pred = regressor.predict(X_test)
     y_pred = sc_y.inverse_transform(y_pred)
     y_pred

     array([ 27.02365867,  25.07154738,  23.08014265,  25.60360588,
             25.7185961 ,  25.51166554,  21.51908439,  20.5263883 ,
             19.92547739,  20.14089559,  20.6917604 ,  20.7563445 ,
             25.08738747,  25.14935289,  27.2315788 ,  27.07049253,
             25.33805772,  24.94547598,  21.68473846,  21.74269855,
             10.85894018,  10.91675138,  29.10373903,  18.50107832,
             19.09365475,  24.11974466,  24.07701629,  24.99330913,
             25.03277914,  22.63982825,  22.47473917,  22.46443094,
             23.69627822,  23.9021053 ,  23.92288917,  22.26477489,
             21.34771325,  21.49237817,  22.87569746,  22.76812804,
             23.00037269,  22.77688125,  22.67370998,  22.62751812,
             22.23479829,  25.86646771,  25.39906014,  21.85260924,
             21.8983412 ,  25.92667878,  26.0507953 ,  20.05506996,
             20.39740941,  23.27591689,  23.39211414,  19.88547393,
             30.39171794,  25.75657245,  21.35005596,  22.27491348,
             22.92848256,  23.14238599,  21.18387918,  21.19548073,
             17.36319321,  18.40306958,  26.60544679,  27.349806  ,
             28.17508458,  32.54126846,  31.63785203,  27.21435078,
             30.5718968 ,  28.06682952,  27.33933515,  28.71027402,
             22.07404903,  12.40640549,  27.57071699,  23.87145198,
             23.90115872,  18.0525907 ,  20.91072574,  20.52050123,
```

Figura 50. Código en Python algoritmo SVM Kernel Poliomial- 2110

Comparamos los valores predichos vs los valores reales del conjunto de datos de prueba (2110), el resultado obtenido es el siguiente:

```
[67] df = pd.DataFrame({'RealValues':sc_y.inverse_transform(y_test.reshape(-1)), 'PredictedValues':y_pred})
     df
```

30	22.0	22.474739
31	22.0	22.464431
32	24.0	23.696278
33	24.0	23.902105
34	24.0	23.922889
35	10.0	22.264775
36	10.0	21.347713
37	10.0	21.492378
38	17.0	22.875697

Figura 51. Predicciones vs. valor real SVM Kernel Polinomial - 2110

Con estos datos evaluamos el rendimiento del modelo bajo la métrica RMSE (error cuadrático medio), el resultado de la evaluación de este tercer modelo usando un kernel polinomial fue de 8.06 el más alto de los 3 modelos desarrollados, por ende, él menos eficiente de esta ronda de pruebas.

```
[69] #calcular RMSE
     sqrt (mean_squared_error (df['RealValues'],df['PredictedValues']))

     8.06833706047319
```

Figura 52. RMSE del modelo SVM Kernel Polinomial- 2110

Continuando con el esquema de experimentación se realiza validación cruzada para dividir el conjunto de datos en:

Tres periodos para entrenamiento y uno para pruebas, así:

- 1510, 1530, 1610 (Entrenamiento)
- 1630 (Pruebas)

```
[ ]  df1510 = df[(df["Ciclo_1510.0"] == 1)]
     df1530 = df[(df["Ciclo_1530.0"] == 1)]
     df1610 = df[(df["Ciclo_1610.0"] == 1)]
     df1630 = df[(df["Ciclo_1630.0"] == 1)]
     #df1710 = df[(df["Ciclo_1710.0"] == 1)]
     #df1730 = df[(df["Ciclo_1730.0"] == 1)]
     #df1810 = df[(df["Ciclo_1810.0"] == 1)]
     #df1830 = df[(df["Ciclo_1830.0"] == 1)]
     #df1910 = df[(df["Ciclo_1910.0"] == 1)]
     #df1930 = df[(df["Ciclo_1930.0"] == 1)]
     #df2010 = df[(df["Ciclo_2010.0"] == 1)]
     #df2030 = df[(df["Ciclo_2030.0"] == 1)]
```

Figura 53. División de datos para Entrenamiento y Pruebas

Se validan los modelos anteriormente desarrollados con los nuevos conjuntos de datos de entrenamiento y pruebas.

<u>Modelo 4</u>: Modelo SVM Radial

```
[ ]  y_pred = regressor.predict(X_test)
     y_pred = sc_y.inverse_transform(y_pred)
     y_pred
```

```
array([15.9935404 , 18.12222492, 13.71106344, 13.72252665, 16.48334115,
       16.464189  , 19.47255976, 19.48796575, 16.45771021, 16.60950962,
       19.42034526, 19.30903929, 19.46962093, 19.45957556, 20.71334934,
       20.7198291 , 20.73481842, 20.83715116, 18.0773725 , 18.03602798,
       23.00807673, 23.16099557, 18.57019336, 18.53875614, 19.11188066,
       14.15752949, 14.10241883, 18.88929002, 18.94954858, 19.377002  ,
       19.41215565, 19.05582995, 15.65747098, 15.63082034, 15.63864416,
       15.66326212, 15.48480478, 20.73242676, 20.80110462, 16.26880993,
```

Figura 54. Predicciones SVM Radial

```
[ ]  df = pd.DataFrame({'RealValues':sc_y.inverse_transform(y_test.reshape(-1)), 'PredictedValues':y_pred})
     df
```

	RealValues	PredictedValues
0	25.0	15.993540
1	24.0	18.122225
2	11.0	13.711063

Figura 55. Predicciones vs. Datos Reales SVM Radial

```
[ ]  #importar las bibliotecas necesarias
     from sklearn.metrics import mean_squared_error
     from math import sqrt

     #calcular RMSE
     sqrt (mean_squared_error (df['RealValues'],df['PredictedValues']))
```

```
3.7981858467178875
```

Figura 56. RMSE del modelo SVM Kernel Radial

Es válido destacar los resultados hallados en esta prueba, al comparar los modelos desarrollados usando un núcleo radial con el primer conjunto de datos de entrenamiento vs. el segundo conjunto de datos de entrenamiento (validación cruzada), se observa que el indicador RMSE mejora en la segunda prueba 3.79 respecto del resultado obtenido en la primera instancia 7.92.

Modelo 5: Modelo SVM Linear

```
[ ]  y_pred = regressor.predict(X_test)
     y_pred = sc_y.inverse_transform(y_pred)
     y_pred

     array([21.58141316, 19.61598137, 19.25289928, 19.25289928, 19.25289928,
            19.25289928, 19.25289928, 19.25289928, 19.25289928, 19.25289928,
            18.69587685, 19.25289928, 19.25289928, 19.25289928, 18.69587685,
            19.25289928, 24.29974971, 24.29974971, 24.29759451, 24.29759451,
            21.86475013, 21.86475013, 21.86475013, 21.86475013, 14.02334238,
            14.02334238, 21.17469928, 21.17469928, 21.17685449, 21.17685449,
            18.22999   , 18.22999   , 18.22999   , 18.22999   , 18.22999   ,
```

Figura 57. Predicciones SVM Linear

```
[ ]  df = pd.DataFrame({'RealValues':sc_y.inverse_transform(y_test.reshape(-1)), 'PredictedValues':y_pred})
     df
```

	RealValues	PredictedValues
0	20.0	21.581413
1	21.0	19.615981
2	19.0	19.252899
3	19.0	19.252899
4	19.0	19.252899
5	19.0	19.252899
6	20.0	19.252899

Figura 58. Predicciones vs. Datos Reales SVM Linear

```
[ ]  #calcular RMSE
     sqrt (mean_squared_error (df['RealValues'],df['PredictedValues']))

     3.638807535923564
```

Figura 59. RMSE del modelo SVM Kernel Linear

Nuevamente se revisan los resultados y se comparan los indicadores de los dos modelos con núcleo linear, el primero tiene un valor RMSE de 6.92 vs. La segunda instancia que como puede observarse en la figura 19 tiene un valor inferior de 3.63, por lo tanto, el mejor desempeño hasta el momento lo tiene el modelo que se entrenó usando los periodos (1510 al 2030).

Modelo 6: Modelo SVM Polinomial:

```
[ ]  y_pred = regressor.predict(X_test)
     y_pred = sc_y.inverse_transform(y_pred)
     y_pred
```

```
array([18.83761536, 21.11713703, 19.61011211, 19.61011211, 19.61011211,
       19.61011211, 19.61011211, 19.61011211, 19.61011211, 19.61011211,
       18.02194636, 19.61011211, 19.61011211, 19.61011211, 18.02194636,
       19.61011211, 23.69732475, 23.69732475, 22.42931825, 22.42931825,
       21.44455638, 21.44455638, 21.44455638, 21.44455638, 18.12026237,
       18.12026237, 21.27430878, 21.27430878, 21.72222287, 21.72222287,
       18.17924044, 18.17924044, 18.17924044, 18.17924044, 18.17924044,
       18.17924044, 17.88929381, 17.88929381, 20.56943267, 20.56943267,
```

Figura 60. Predicciones SVM Polinomial

```
[ ]  df = pd.DataFrame({'RealValues':sc_y.inverse_transform(y_test.reshape(-1)), 'PredictedValues':y_pred})
     df
```

	RealValues	PredictedValues
0	20.0	18.837615
1	21.0	21.117137
2	19.0	19.610112
3	19.0	19.610112
4	19.0	19.610112
5	19.0	19.610112

Figura 61. Predicciones vs. Datos Reales SVM Polinomial

```
[ ]  #calcular RMSE
     sqrt (mean_squared_error (df['RealValues'],df['PredictedValues']))
```

```
4.062237003136947
```

Figura 62. RMSE del modelo SVM Kernel Polinomial

De la comparación entre los modelos polinomiales desarrollados previamente con los diferentes conjuntos de entrenamiento, este presenta un mejor rendimiento del indicador RMSE, en donde se observa 4.06 respecto de los 8.06 del modelo entrenado inicialmente.

Continuando con el esquema de experimentación se realizan cambios en la selección de las variables del conjunto de entrenamiento, *se eliminan los días y se agrupan algunos elementos para dejar principalmente el número de inscritos por asignatura y periodo*, Cabe resaltar que para este punto del desarrollo se involucran variables nuevas con base en el diseño de un *Grafo dirigido*, se dispone del uso de la librería *NetworkX de Python* para modelar el plan de estudios del programa de Ing. de Sistemas, se adicionan nodos, los cuales corresponden a los Id de las asignaturas que lo componen y enlaces (Edge) los cuales corresponden a las dependencias entre asignaturas, es importante destacar que estas dependencias se encuentran estructuradas con base plan de estudios del programa. Los detalles de la transformación y adición de estas nuevas variables se pueden encontrar en el ‘Capítulo 3.2 Limpieza y construcción de datos’.

```
[ ] df.dtypes

Sesion               object
IDCurso              object
Ciclo                object
TipoClase            object
Componente           object
NroSemanas            int64
Retiros               int64
Dias                  int64
NroHoras              int64
HorasSemanales        int64
TotalInscripciones    int64
dtype: object
```

Figura 63. Variables nuevas, Nuevo conjunto de datos.

A continuación, se relacionan los detalles de la división y los resultados sobre el nuevo conjunto de datos para entrenamiento y pruebas:

- Entrenamiento (1510, 15030 y 1610)
- Pruebas (1630)

Primer conjunto de datos de entrenamiento (1510, 15030 y 1610) y Primer conjunto de datos de Pruebas (1630).

Modelo 7: Modelo SVM radial

```
[64] from sklearn.svm import SVR
     regressor = SVR(kernel = 'rbf')
     regressor.fit(X_train, y_train)

SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

Figura 64. Modelo SVM Kernel radial

Dado que esta prueba se realizó con base en un nuevo conjunto de datos, es necesario revisar los resultados sobre cada uno de los núcleos del SVM, como se observa en la imagen previa se entrena un nuevo modelo radial, a continuación, se presentan los resultados obtenidos:

```
[90] y_pred = regressor.predict(X_test)
     y_pred = sc_y.inverse_transform(y_pred)
     y_pred

array([ 28.58258771,  59.67603944,  93.73386005,  36.16100865,
       176.95047066,  47.46256537,  45.83148428,  41.94209739,
        43.51276389,  41.02237041,  33.34056592,  36.09672455,
        30.45911573,  35.43452914,  43.54074412,  56.14176259,
        85.67752133,  33.15982194,  50.70896146,  52.62345783,
        49.72236335,  44.7514914 ,  55.37433016,  38.28799705,
        36.40953112,  53.05614184,  61.0742077 ,  51.71682412,
        49.64582763,  54.17012511,  30.32725778,  41.79265957,
        41.06328029, 182.37539277, 204.57333363, 195.92557789,
       172.37325446, 274.47064175,  31.78334544,  65.91262732,
        40.28786091,  39.97654768,  25.06455566,  37.85307475,
        22.49791053, 210.96959003,  65.03599355,  20.86184224,
        30.44205096,  30.83334717,  45.98815689, 285.24796312])
```

Figura 65. Predicciones SVM Kernel radial – 1630

```
[91] df = pd.DataFrame({'RealValues':sc_y.inverse_transform(y_test.reshape(-1)), 'PredictedValues':y_pred})
     df
```

	RealValues	PredictedValues
0	49.0	28.582588
1	0.0	59.676039
2	106.0	93.733860
3	26.0	36.161009
4	218.0	176.950471
5	20.0	47.462565
6	98.0	45.831484
7	98.0	41.942097
8	98.0	43.512764

Figura 66. Predicciones vs. datos reales SVM Kernel radial – 1630

```
#importar las bibliotecas necesarias
from sklearn.metrics import mean_squared_error
from math import sqrt

#calcular RMSE
sqrt (mean_squared_error (df['RealValues'],df['PredictedValues']))
```

```
35.99715104775627
```

Figura 67. RMSE del modelo SVM Kernel radial - 1630

Los resultados sobre el nuevo conjunto de entrenamiento y pruebas para un kernel radial difieren en gran medida con los anteriores modelos, debido a la agrupación de los datos, esta vez el número de inscritos se calculó por asignatura y por periodo, anteriormente esta cifra se calculaba por modelo de reunión, es decir, por día para cada clase. Es necesario revisar el resultado con los demás kernels.

Modelo 8: SVM linear

```
[93] #Probamos usando un kernel Linear
     from sklearn.svm import SVR
     regressor = SVR(kernel='linear')
     regressor.fit(X_train, y_train)
```

```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='linear', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

Figura 68. Modelo SVM Kernel linear

```
[94] y_pred = regressor.predict(X_test)
     y_pred = sc_y.inverse_transform(y_pred)
     y_pred
```

```
array([ 12.17052685,  70.52468528,  76.29720993,  20.486433  ,
       178.15661796,  31.21044819,  37.0996858 ,  31.61383524,
        32.41571804,  30.4501472 ,  36.58375699,  26.5207469 ,
        18.69463925,  29.55648622,  32.34391788,  56.68216017,
        59.39449202,  15.77791698,  29.70178142,  42.28526121,
        36.79941065,  40.92158725,  42.34335243,  33.12330979,
        27.14171239,  73.64400345,  49.02796739,  48.04454161,
        47.82255372,  34.42468194,  16.73937318,  21.39734215,
        41.08581318, 288.74833278, 281.12919435, 280.30967286,
       287.60100271, 275.53834659,  23.88503773,  42.60367607,
        39.66764442,  26.02613206,  15.41179162,  24.33856533,
```

Figura 69. Predicciones SVM Kernel linear – 1630

```
[95] df = pd.DataFrame({'RealValues':sc_y.inverse_transform(y_test.reshape(-1)), 'PredictedValues':y_pred})
     df
```

	RealValues	PredictedValues
0	49.0	12.170527
1	0.0	70.524685
2	106.0	76.297210
3	26.0	20.486433
4	218.0	178.156618

Figura 70. Predicciones SVM Kernel linear – 1630

```
[96] #calcular RMSE
     sqrt (mean_squared_error (df['RealValues'],df['PredictedValues']))

     29.165800868148082
```

Figura 71. RMSE del modelo SVM Kernel linear – 1630

Modelo *9:* SVM Polinomial

```
[98] #Probamos usando un kernel Polynomial
     from sklearn.svm import SVR
     regressor = SVR(kernel='poly', C=5, gamma=0.05, degree=2)
     regressor.fit(X_train, y_train)

     SVR(C=5, cache_size=200, coef0=0.0, degree=2, epsilon=0.1, gamma=0.05,
         kernel='poly', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

Figura 72. Modelo SVM Kernel Polinomial Modelo 9

```
[99] y_pred = regressor.predict(X_test)
     y_pred = sc_y.inverse_transform(y_pred)
     y_pred

     array([ 34.51203754,  73.88531321,  98.47807983,  35.89650582,
            220.81672599,  49.04883222,  62.13405534,  57.82931129,
             62.16081859,  46.31208237,  50.47868032,  57.24018607,
             36.55186819,  52.68996173,  58.51385062, 107.89721302,
             70.77751075,  33.93840374,  29.66832879,  74.1285407 ,
             63.32356277,  61.73334927,  41.91089516,  57.82099792,
             51.57799456,  73.38136058,  76.6332173 ,  68.75417872,
             68.38141199,  57.87285298,  28.76037742,  45.20847827,
             57.12458365, 161.74078381, 187.60477655, 177.22592719,
            124.29377226, 255.81006168,  21.36871803,  65.00760291,
             56.58859673,  32.09887057,  39.78723886,  32.86954913,
             21.00742493, 272.15406107,  82.66942924, -35.44298095,
             28.5092512 ,  26.75964876,  51.27559225, 297.76371765])
```

Figura 73. Predicciones SVM Kernel Polinomial Modelo 9

```
[101] #calcular RMSE
      sqrt (mean_squared_error (df['RealValues'],df['PredictedValues']))

      44.72395712018452
```

Figura 74. RMSE del modelo SVM Kernel Polinomial – Modelo 9

Con base en el mismo conjunto de datos para entrenamiento y pruebas se implementa el uso de un modelo basado en Redes Neuronales, a continuación, se presenta el desarrollo y los resultados:

Modelo 10: Disponemos del uso del Python y de la librería "*Keras"* para diseñar las diferentes estructuras de las redes neuronales que se implementarán, en primer lugar, normalizamos los conjuntos de entrenamiento y pruebas usando la medida de Z-Score y posteriormente definimos la función de la red neuronal como "regresión", a continuación, se relacionan las líneas de código usado para este fin:

```
[87] # Z-SCORE
     from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     X_train = sc.fit_transform(X_train)
     X_test = sc.transform(X_test)
```

```
[88] # Definición de la Red Neuronal para regresión

     from keras.models import Sequential
     from keras.layers import Dense
     from keras import backend
```

```
[89] # Definición del Error Cuadrado Medio como la función métrica

     def rmse(y_true, y_pred):
       return backend.sqrt(backend.mean(backend.square(y_pred - y_true), axis=-1))
```

Figura 75. Diseño modelo RN

El diseño del primer modelo de RN considera el uso de una sola capa intermedia con 50 neuronas y una capa de salida, la capa de entrada debe tener el mismo número de neuronas que la cantidad de variables de los datos de entrenamiento después de usar el algoritmo de One Hot Encoding para numérizar las variables categóricas.

```
[ ]  # Construcción del primer modelo: una capa oculta con 100 nodos

     model1 = Sequential()

     # Capa de entrada y Capa oculta - con neurona de sesgo
     model1.add(Dense(50,input_dim=113, use_bias=True, activation = 'sigmoid'))

     # Capa de salida
     model1.add(Dense(1, use_bias=True, activation='linear'))
```

Figura 76. Modelo 10 RN

La estructura del primer modelo diseñado usando redes neuronales tiene una capa de entrada con 113 neuronas que se activan usando una función Sigmoidal, una capa intermedia con 50 neuronas y una capa de salida de una sola neurona, a continuación, se describe gráficamente esta distribución:

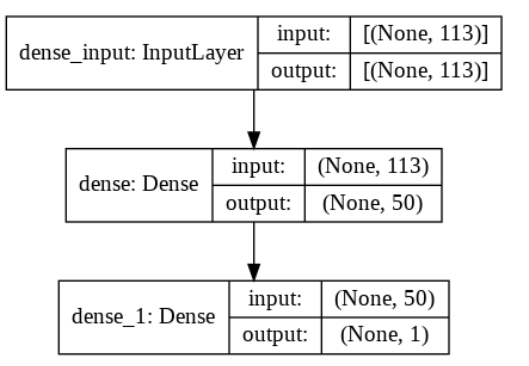


Figura 77. Estructura modelo 10 RN

El modelo 10 se compila usando 300 épocas con un Batch Size de 32 registros para calcular la gradiente y una tasa de aprendizaje de 0.015, lo cual modifica el algoritmo original de optimización, para este caso puntual se seleccionó el algoritmo "Adam". Como se indicó previamente la métrica de evaluación seleccionada fue el RMSE, por lo tanto, se espera que esta cifra disminuya conforme avanzan las épocas, lo cual al final podría resultar en un modelo sobreajustado, para determinar esto es necesario revisar el resultado del modelo en el dataset de entrenamiento como en el dataset de pruebas.

```
[ ]  import tensorflow.python.keras.optimizers #modificado del original

     # Optimizar , Compilar y Entrenar el Modelo
     from numpy.random import seed

     seed(1)
     opt =tensorflow.keras.optimizers.Adam(learning_rate=0.015) #modificado del original

     model1.compile(optimizer=opt,loss='mean_squared_error',metrics=[rmse])
     history = model1.fit(X_train,y_train,epochs = 300, batch_size=32,validation_split=0.1)

     print(model1.summary())
```

```
Epoch 278/300
4/4 [==============================] - 0s 10ms/step - loss: 1457.0819 - rmse: 20.3395 - val_loss: 9927.9004 - val_rmse: 52.6404
Epoch 279/300
4/4 [==============================] - 0s 10ms/step - loss: 1444.9617 - rmse: 20.2623 - val_loss: 9969.3066 - val_rmse: 52.8013
```

Figura 78. Compilación Modelo 10 RN

Las métricas de desempeño del modelo 10 tanto en el conjunto de datos de entrenamiento con en el de pruebas fueron las siguientes:

- Desempeño en Entrenamiento:
- Coeficiente de determinación: 0.804601204338104
- Coeficiente de Correlación: 0.8969956545815059
- MSE: 48.29430108272599

- Desempeño en Pruebas:
- Coeficiente de determinación: 0.5254213175208764
- Coeficiente de Correlación: 0.7248595157138219
- MSE: 60.89433334482907

El desempeño del modelo 10 con el dataset de pruebas no es eficiente si lo comparamos con los resultados obtenidos con el mismo conjunto de datos sobre el modelo 8 SVM Linear, el cual tuvo un RMSE de *29.16* vs los *60.89* del modelo 10. A continuación se describe el desempeño del RMSE por época junto con la función de perdida. Cabe señalar que el rendimiento óptimo se da sobre las 140 épocas, después de esto la eficiencia disminuye.

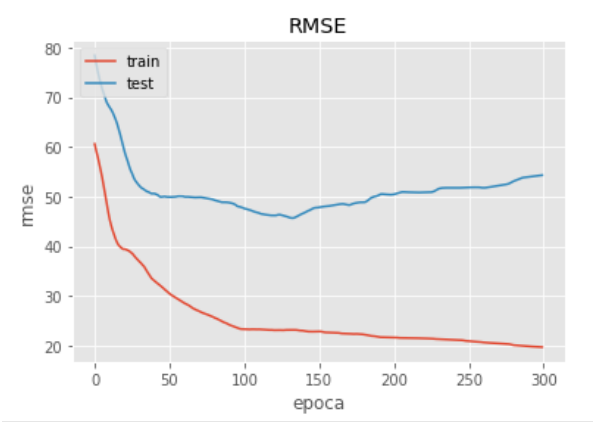


Figura 79. RMSE por época Modelo 10

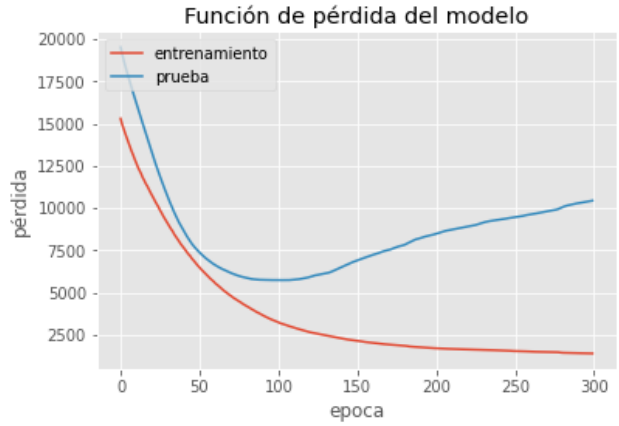


Figura 80. Función de perdida por época Modelo 10

Atendiendo a las anteriores observaciones para los siguientes modelos se reduce el número de épocas, es necesario revisar el rendimiento y ajustar los parámetros en los esquemas de experimentación.

Modelo 11: La estructura de este modelo se ajusta con base en los aprendizajes obtenidos del modelo anterior, se configura un modelo con una capa inicial de 113 neuronas, con dos capas intermedias de 50 neuronas cada una, se adiciona una neurona de sesgo “siempre activa” en cada una de las capas intermedias, nuevamente se deja una sola neurona en la capa de salida con una función de activación “linear”. Esta vez se establecen solamente 150 épocas se usa un Bacht Size de 1 para calcular la gradiente y una taza de aprendizaje de 0.015, los resultados son los siguientes:

- Desempeño en Entrenamiento:
- Coeficiente de determinación: 0.9896291054675725

- Coeficiente de correlación: 0.9948010381315314
- MSE: 11.126102404302738

- Desempeño en Pruebas:
- Coeficiente de determinación: 0.5836066731493661
- Coeficiente de correlación: 0.7639415377824184
- MSE: 57.03936047485954

El indicador RMSE mejoro respecto del modelo 10, pero aún su eficiencia es inferior a las de los modelos SMV, por lo cual es conveniente continuar con el esquema de experimentación.

```
[ ]  # Construcción de un segundo modelo: dos capas ocultas con 5 nodos

     model2=Sequential()

     # Capa de entrada y Primera capa oculta - con neurona de sesgo
     model2.add(Dense(50, input_dim=113, use_bias=True, activation = 'sigmoid'))

     # Segunda capa oculta - con neurona de sesgo
     model2.add(Dense(50, use_bias=True, activation='sigmoid'))

     # Capa de salida
     model2.add(Dense(1,activation='linear'))
```

Figura 81. Modelo 11 RN

Figura 82. RMSE por época Modelo 11

Modelo 12: Este modelo presenta una variación respecto de los dos anteriores, se usan 3 capas ocultas cada una con 50 neuronas y con activación Sigmoidal y una capa de salida con una neurona con una función de activación linear, nuevamente se compila con 300 épocas y con un Bacht Size de 32, los resultados conseguidos son mejores que los de los modelos previos. Es importante destacar en primer lugar que los coeficientes de correlación aumentaron significativamente, en segundo lugar, que no se presenta sobre ajuste debido a que el indicador MSE es similar tanto en entrenamiento como en pruebas y en tercer lugar que después de las 250 épocas el modelo se estabiliza debido a que el indicar MSE se mantiene hasta el final. La eficiencia es similar en comparación con los modelos SMV diseñados previamente.

- Desempeño en Entrenamiento:
- Coeficiente de determinación: 0.9354612825429054
- Coeficiente de correlación: 0.9671924744035726+0j
- MSE: 27.755247816104795

- Desempeño en Pruebas:
- Coeficiente de determinación: 0.9066112375260632
- Coeficiente de correlación: 0.9521613505735587+0j
- MSE: 27.01282949713059

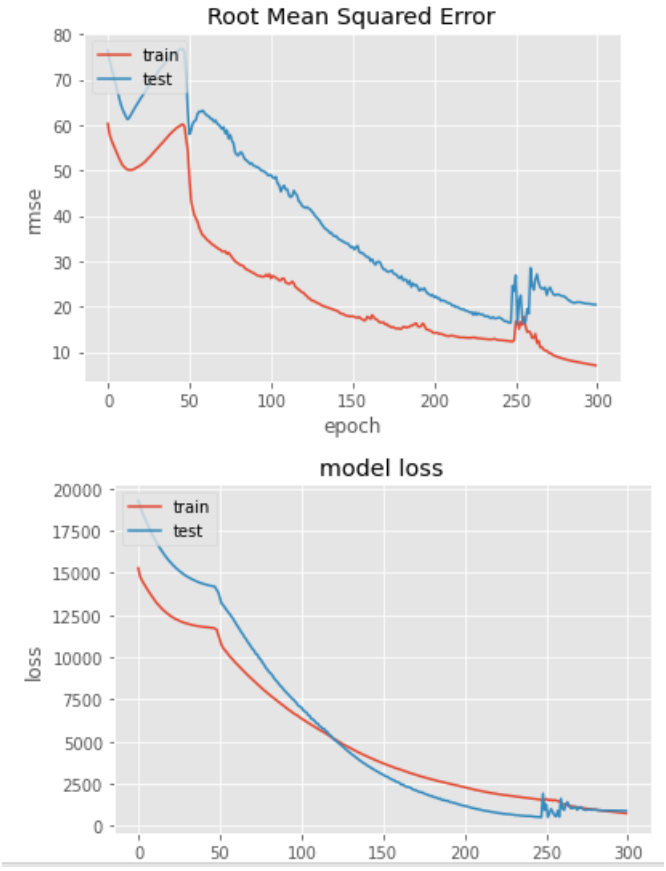


Figura 83. RMSE por época Modelo 12

Segundo conjunto de datos de entrenamiento (1630, 1710 y 1730) y Segundo conjunto de datos de Pruebas (1810):

Debido a que contamos con un nuevo conjunto de datos para entrenamiento y pruebas, revisamos el comportamiento de los modelos SVM y RN para validar su desempeño.

Modelo 13: Este modelo es del tipo SVM y usa un núcleo linear, probamos su desempeño en el dataset de pruebas, encontramos que el indicador RMSE es de 29.71.

```
[ ] from sklearn.svm import SVR
    regressor = SVR(kernel = 'rbf')
    regressor.fit(X_train, y_train)

    SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
        kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)

[ ] y_pred = regressor.predict(X_test)
    y_pred = sc_y.inverse_transform(y_pred)
    y_pred

    array([ 41.13958334,  38.38451669, 105.71615597,  33.36488209,
           209.5424506 ,  25.41800471,  91.32640423,  31.83984016,
            39.32415913,  39.65933529,  29.80405886,  36.70643197,
            48.31248475,  36.22843475,  47.62703312,  38.36480941,
            28.87910491,  54.25765726,  25.34428989,  49.64987992,
            37.86958781,  44.13217437,  37.45020438,  31.23268428,
            41.3840825 ,  26.84905539,  33.00096479,  33.85103753,
           262.89583553,  65.35140397,  62.64700473,  40.11385824,
            37.92632899,  31.47033018,  14.31371565,  24.27448699,
           270.65820714,  33.64017634,   9.58372696,  28.24227754,
            26.96203632,  30.08114664, 273.1716092 ,  40.32177363])
```

Figura 84. SVM Radial - modelo 13

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

RealValues	PredictedValues
43.0	41.13
0.0	38.38
103.0	105.71
35.0	33.36
204.0	209.54

Tabla 2. Predicciones Modelo 13

Modelo 14: Este modelo es del tipo SVM y usa un núcleo linear, probamos su desempeño en el dataset de pruebas, encontramos que el indicador RMSE es de 27.31.

```
[ ]  #Probamos usando un kernel Linear
     from sklearn.svm import SVR
     regressor = SVR(kernel='linear')
     regressor.fit(X_train, y_train)
```

```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='linear', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

```
[ ]  y_pred = regressor.predict(X_test)
     y_pred = sc_y.inverse_transform(y_pred)
     y_pred
```

```
array([ 38.10216   ,  44.38408338, 107.03485966,  31.41432609,
       234.72397406,  16.87224598,  70.69382014,  24.75467664,
        47.09723493,  44.1493386 ,  35.07960824,  26.63870035,
        28.90247751,  29.90681371,  71.21981057,  46.05421451,
        22.61984395,  73.53351194,  19.45578322,  44.23523244,
        37.2750067 ,  46.55107347,  41.22570595,  21.72766171,
        32.03377217,  18.48364042,  31.22104802,  52.81742708,
       273.7541274 ,  57.33509498,  64.58405355,  32.96833161,
        35.09843162,  12.42745186,   4.19784686,  21.38399998,
       295.59909343,  24.60176894,  -6.72811176,  18.20533116,
        20.41842388,  30.30534868, 340.58937082,  41.24587968])
```

Figura 85. SVM Linear modelo 13

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

RealValues	PredictedValues
43.0	38.10
0.0	44.38
103.0	107.03
35.0	31.41
204.0	234.72
35.0	16.87

Tabla 3. Predicciones Modelo 14

Modelo 15*:* Este modelo es del tipo SVM y usa un núcleo Polinomial, probamos su desempeño en el dataset de pruebas, encontramos que el indicador RMSE es de 26.86.

```
[ ]  #Probamos usando un kernel Polynomial
     from sklearn.svm import SVR
     regressor = SVR(kernel='poly', C=5, gamma=0.05, degree=2)
     regressor.fit(X_train, y_train)

     SVR(C=5, cache_size=200, coef0=0.0, degree=2, epsilon=0.1, gamma=0.05,
         kernel='poly', max_iter=-1, shrinking=True, tol=0.001, verbose=False)

[ ]  y_pred = regressor.predict(X_test)
     y_pred = sc_y.inverse_transform(y_pred)
     y_pred

     array([ 48.03412647,  48.68823273,  98.77355112,  35.39989024,
            198.56597851,  28.33997102,  72.07466998,  46.80987893,
             44.67027904,  44.0865363 ,  46.01867321,  35.17773742,
             43.83873521,  41.69811574,  53.90376724,  51.54536306,
             30.22096555,  60.07362339,  20.07421529,  50.66675536,
             48.53915797,  46.84849764,  48.18530694,  18.26917913,
             41.71385101,  26.90628265,  36.40555946,  49.19533659,
            323.95583792,  62.19221711,  73.50720819,  50.17066903,
             48.53934439,  43.19793372,  12.84548492,  37.23608891,
            375.79527179,  31.55915371, -28.66296641,  26.60738102,
             29.35759857,  36.52912312, 300.58299705,  54.91011339])
```

Figura 86. SVM Polinomial - modelo 15

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

RealValues	PredictedValues
43.0	48.03
0.0	48.68
103.0	98.77
35.0	35.39
204.0	198.56

Figura 87. Predicciones Modelo 15

Modelo 16*:* Este modelo es del tipo RN (Redes Neuronales) y se diseñó usando los siguientes parámetros:

- Capa de entrada: 111 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- Capa oculta: 50 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- Capa de Salida: 1 Neurona, Función de activación (Linear), Neurona de Sesgo (Verdadero).
- Batch Size: 32.
- Learning_rate=0.015
- Epocas: 300

El desempeño de este modelo es el siguiente:
Desempeño en entrenamiento

- Coeficiente de determinación 0.7958150345108024
- Coeficiente de correlación: 0.8920846565829963
- Mse: 39.03321493442109

Desempeño en Pruebas

- Coeficiente de determinación: 0.7122181464368749
- Coeficiente de correlación: 0.8439301786503874
- Mse: 45.6325349893067

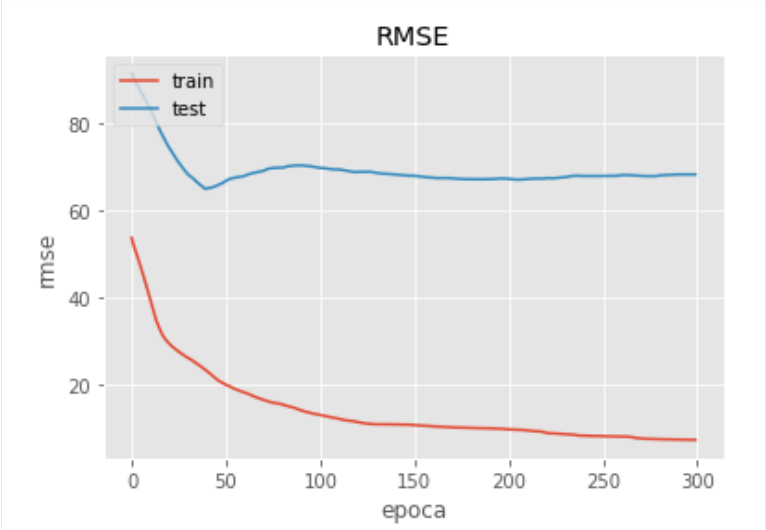


Figura 88. RN RMSE - modelo 16

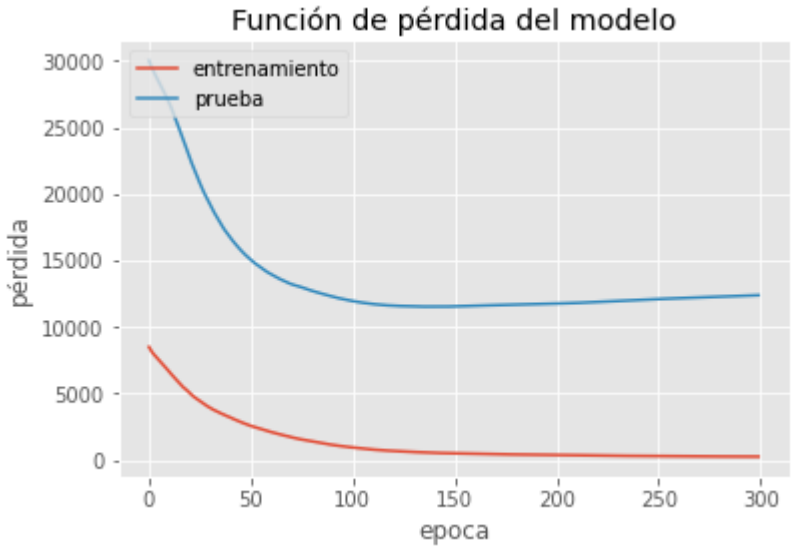


Figura 89. RN Función de Perdida - modelo 16

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

Valor Real	Valor Predicho
43.0	46.72
0.0	1.78
103.0	105.92
35.0	26.75
204.0	221.58

Tabla 4. Predicciones Modelo 16

Modelo 17*:* Este modelo es del tipo RN (Redes Neuronales) y se diseñó usando los siguientes parámetros:

- Capa de entrada: 111 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- 2 capas ocultas: 50 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- Capa de Salida: 1 Neurona, Función de activación (Linear), Neurona de Sesgo (Verdadero).
- Batch Size: 1.
- Learning_rate=0.015
- Epocas: 150

El desempeño de este modelo es el siguiente:

Desempeño en entrenamiento:

- Coeficiente de determinación: 0.8972499097465817
- Coeficiente de correlación: 0.9472327642911121
- MSE: 27.689400214465387

Desempeño en Pruebas

- Coeficiente de determinación: 0.9237448571917611
- Coeficiente de correlación: 0.9611164639063058
- MSE: 23.489699561930912

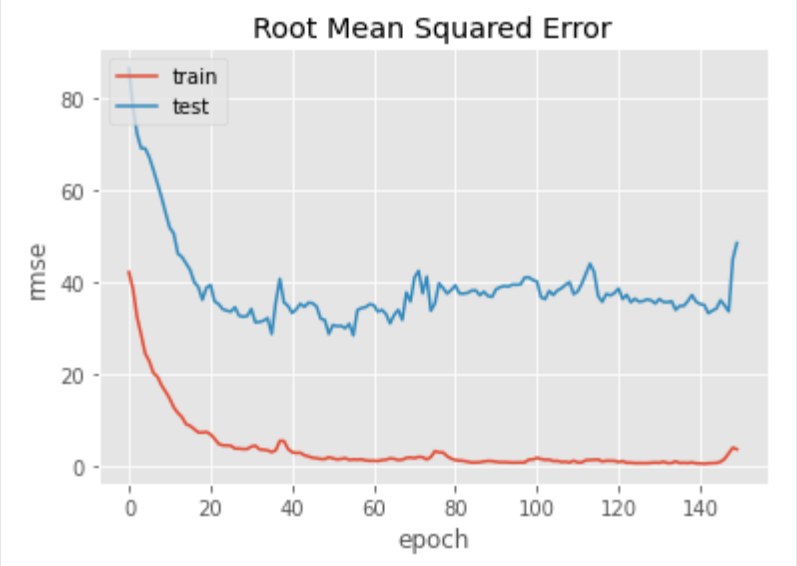


Figura 90. RN RMSE - modelo 17

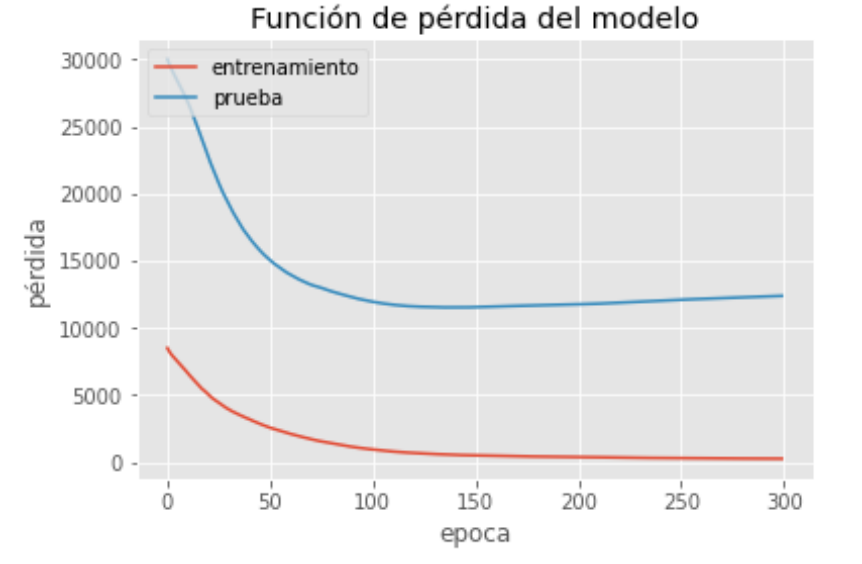


Figura 91. RN Función de Perdida - modelo 17

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

Valor Real	Valor Predicho
43.0	41.39
0.0	9.24
103.0	97.11
35.0	28.40
204.0	219.22

Tabla 5. Predicciones Modelo 17

Modelo 18*:* Este modelo es del tipo RN (Redes Neuronales) y se diseñó usando los siguientes parámetros:

- Capa de entrada: 111 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- 3 capas ocultas: 50 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- Capa de Salida: 1 Neurona, Función de activación (Linear), Neurona de Sesgo (Verdadero).
- Batch Size: 32.
- Learning_rate=0.015
- Epocas: 300

El desempeño de este modelo es el siguiente:

Desempeño en entrenamiento

- Coeficiente de determinación: 0.9272507629913733
- Coeficiente de correlación: 0.9629386081113236+0j
- MSE: 23.29896687395755

Desempeño en Pruebas

- Coeficiente de determinación: 0.89218388114565
- Coeficiente de correlación: 0.9445548587274589+0j
- MSE: 27.930887771260227

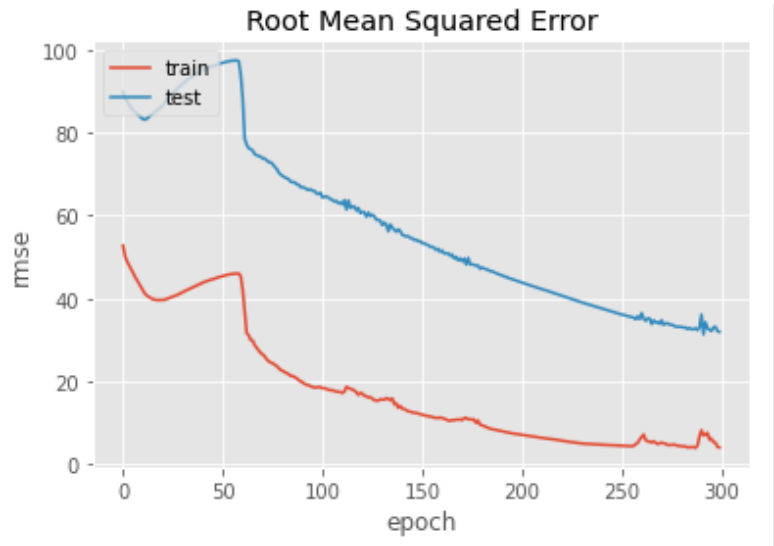


Figura 92. RN RMSE - modelo 18

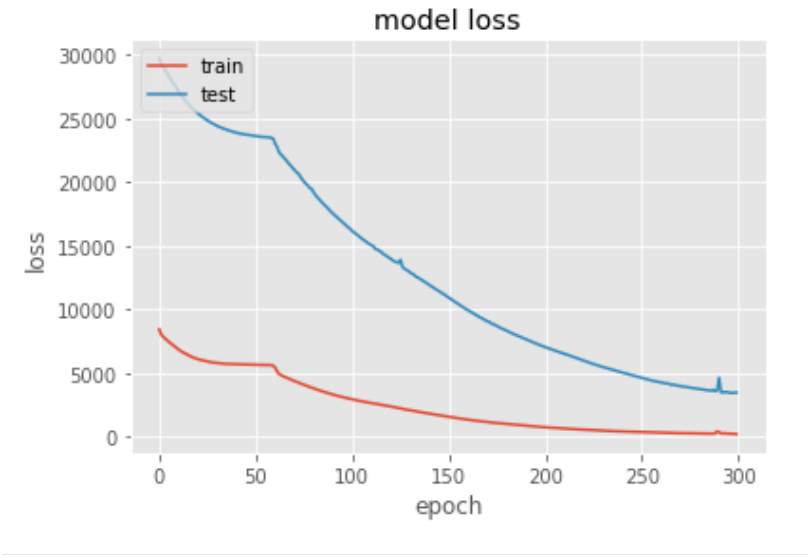


Figura 93. RN Función de Perdida - modelo 18

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

Valor Real	Valor Predicho
43.0	41.79
0.0	-0.22
103.0	107.44
35.0	26.18
204.0	208.90

Tabla 6. Predicciones Modelo 18

Tercer conjunto de datos de entrenamiento (1810, 1830, 1910) y Tercer conjunto de datos de Pruebas (1930):

Modelo 19*:* Este modelo es del tipo SVM y usa un núcleo radial, probamos su desempeño en el dataset de pruebas, encontramos que el indicador RMSE es de 47.18.

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

RealValues	PredictedValues
41.0	46.47
119.0	79.44
22.0	29.57
27.0	30.38
0.0	28.84

Tabla 7. Predicciones Modelo 19

Modelo 20: Este modelo es del tipo SVM y usa un núcleo linear, probamos su desempeño en el dataset de pruebas, encontramos que el indicador RMSE es de 23.86.

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

RealValues	PredictedValues
41.0	44.87
119.0	91.05
22.0	27.80
27.0	32.52
0.0	27.32

Tabla 8. Predicciones Modelo 20

Modelo 21: Este modelo es del tipo SVM y usa un núcleo polinomial, probamos su desempeño en el dataset de pruebas, encontramos que el indicador RMSE es de 28.66.

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

RealValues	PredictedValues
41.0	34.90
119.0	90.87
22.0	39.25
27.0	21.50
0.0	37.62

Tabla 9. Predicciones Modelo 21

Modelo 22: Este modelo es del tipo RN (Redes Neuronales) y se diseñó usando los siguientes parámetros:

- Capa de entrada: 111 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- 1 capa oculta: 50 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- Capa de Salida: 1 Neurona, Función de activación (Linear), Neurona de Sesgo (Verdadero).
- Batch Size: 32.
- Learning_rate=0.015
- Epocas: 300

El desempeño de este modelo es el siguiente:
Desempeño en entrenamiento:

- Coeficiente de determinación: 0.6908801713449766
- Coeficiente de correlación: 0.8311920183357012
- MSE: 48.618505938602574

Desempeño en Prueba

- Coeficiente de determinación: 0.6080958007387118
- Coeficiente de correlación: 0.779804976092556
- MSE: 51.81522084595333

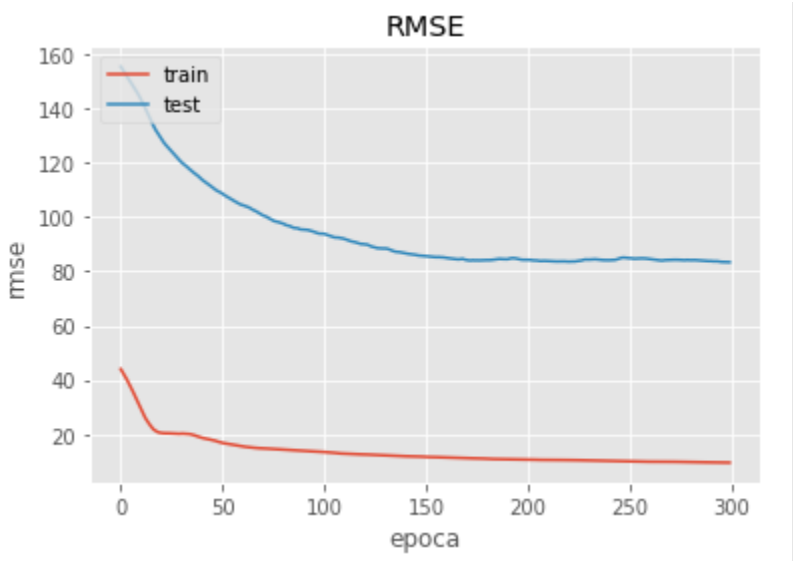


Figura 94. RN RMSE - modelo 22

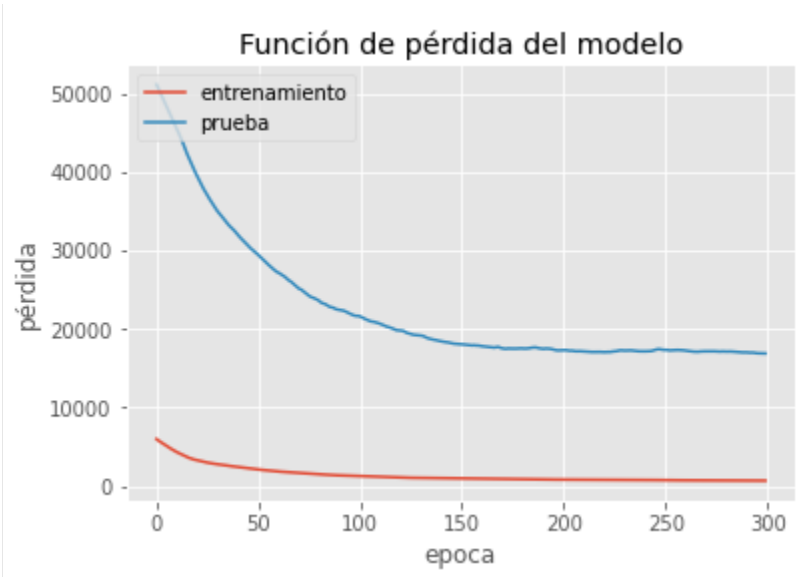


Figura 95. RN Función de Perdida - modelo 22

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

Valor Real	Valor Predicho
41.0	63.33
119.0	100.59
22.0	79.55
27.0	21.26
0.0	82.28

Tabla 10. Predicciones Modelo 22

Modelo 23: *Este modelo es del tipo RN (Redes Neuronales) y se diseñó usando los siguientes parámetros:*

- Capa de entrada: 111 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- 2 capas ocultas: 50 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- Capa de Salida: 1 Neurona, Función de activación (Linear), Neurona de Sesgo (Verdadero).
- Batch Size: 1.
- Learning_rate=0.015
- Epocas: 150

El desempeño de este modelo es el siguiente:

Desempeño en entrenamiento:

- Coeficiente de determinación: 0.5211432165281777
- Coeficiente de correlación: 0.7219024979373445
- MSE: 60.511929154962225

Desempeño en prueba:

- Coeficiente de determinación: 0.7535962731450305
- Coeficiente de correlación: 0.8680992300106195
- MSE: 41.085732210564046

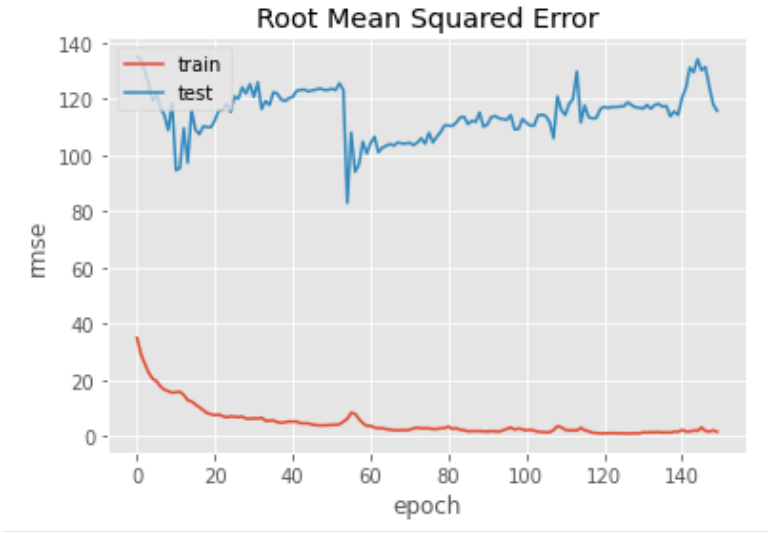


Figura 96. RN RMSE - modelo 23

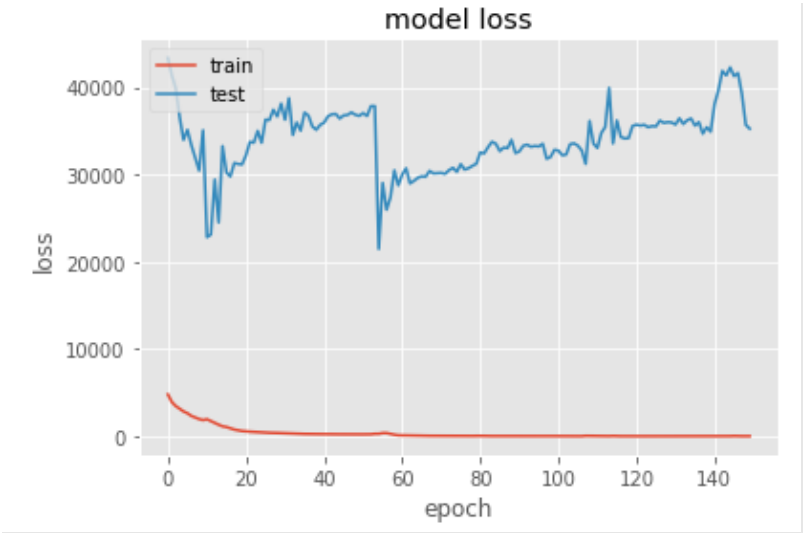


Figura 97. RN Función de Perdida - modelo 23

Modelo 24: Este modelo es del tipo RN (Redes Neuronales) y se diseñó usando los siguientes parámetros:

- Capa de entrada: 111 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- 3capas ocultas: 50 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- Capa de Salida: 1 Neurona, Función de activación (Linear), Neurona de Sesgo (Verdadero).
- Batch Size: 32.
- Learning_rate=0.015
- Epocas: 300

El desempeño de este modelo es el siguiente:

Desempeño en entrenamiento:

- Coeficiente de determinación: 0.3871839245458599
- Coeficiente de correlación: 0.62224105019346+0j
- MSE: 68.4546968166241

Desempeño en prueba:

- Coeficiente de determinación: 0.6390787861921035
- Coeficiente de correlación: 0.7994240340345689+0j
- MSE: 49.724863507753625

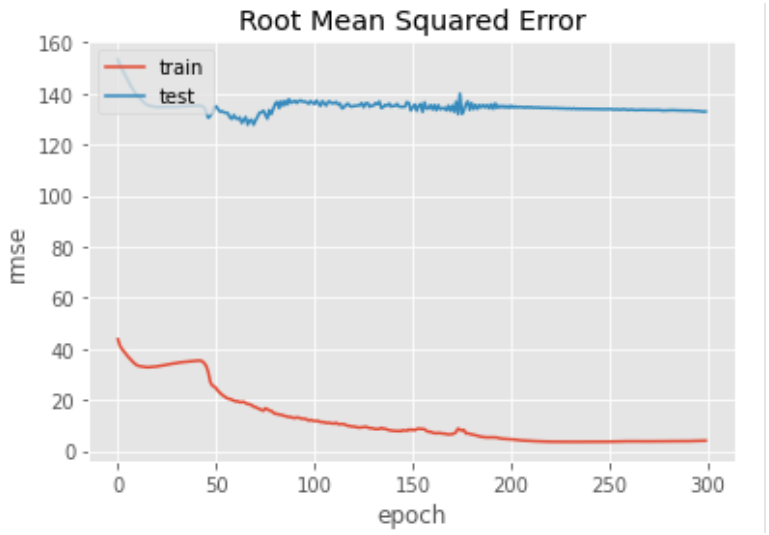


Figura 98. RN RMSE - modelo 24

Figura 99. RN Función de Perdida - modelo 24

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

Valor Real	Valor Predicho
41.0	41.99
119.0	99.10
22.0	-0.03
27.0	25.48
0.0	0.17

Tabla 11. Predicciones Modelo 24

Cuarto conjunto de datos de entrenamiento (1930, 2010, 2030) y Cuarto conjunto de datos de Pruebas (2110), es importante señalar que este conjunto de datos es atípico debido a que se da en periodo de pandemia, a continuación, se relacionan los resultados obtenidos:

Modelo 25: Este modelo es del tipo SVM y usa un núcleo radial, probamos su desempeño en el dataset de pruebas, encontramos que el indicador RMSE es de 94.73.

```
[90] from sklearn.svm import SVR
     regressor = SVR(kernel = 'rbf')
     regressor.fit(X_train, y_train)

     SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
         kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)

[91] y_pred = regressor.predict(X_test)
     y_pred = sc_y.inverse_transform(y_pred)
     y_pred

     array([ 56.73110689,  45.20440453,  94.80772214, 125.49438356,
             47.62055501,  73.97185664, 239.51994221,  50.36112655,
             25.20361085, 109.54506454,  20.32830167,  58.27894238,
              8.26777885,  65.23501323,  77.24267324,  56.66089395,
             20.91691401,  71.13630875,  21.48030467,  78.70727368,
             57.77627696,  41.36393586,  29.48790272,  57.80066706,
             18.99140976,  44.91560257,  16.58330842,  63.98089908,
             26.48254756,  15.79676623, 420.66159583,  41.1699756 ,
             51.44933056,  22.70876206,  31.26055609,  57.52623646,
```

Figura 100. SVM Radial modelo 25

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

RealValues	PredictedValues
68.0	56.73
0.0	45.20
87.0	94.80
87.0	125.49
0.0	47.62

Tabla 12. Predicciones Modelo 25

Modelo 26: Este modelo es del tipo SVM y usa un núcleo linear, probamos su desempeño en el dataset de pruebas, encontramos que el indicador RMSE es de 111.66.

```
[94] #Probamos usando un kernel Linear
     from sklearn.svm import SVR
     regressor = SVR(kernel='linear')
     regressor.fit(X_train, y_train)

     SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
         kernel='linear', max_iter=-1, shrinking=True, tol=0.001, verbose=False)

[95] y_pred = regressor.predict(X_test)
     y_pred = sc_y.inverse_transform(y_pred)
     y_pred

     array([ 6.32056621e+01,  6.68558571e+01,  1.62345844e+02,  1.35105113e+02,
             3.18014285e+01,  4.90734525e+01,  3.09341621e+02,  5.90568087e+01,
             3.32530305e+01,  5.12293642e+01,  2.59636810e+01,  6.70800662e+01,
             1.34044566e+01,  6.98377277e+01,  9.65310254e+01,  5.50037346e+01,
             2.54278468e+01,  4.24463557e+01,  3.82803278e+01,  9.58593076e+01,
             7.64304840e+01,  2.56488184e+01,  2.10157604e+01,  8.06834530e+01,
             3.07639526e+01,  5.34507307e+01,  3.58161774e+01,  3.80231431e+01,
             1.95967802e+01,  1.33427865e+01,  7.02560031e+02,  2.62407129e+01,
             1.70465978e+01, -5.32067397e-01,  1.81978732e+01,  5.98680028e+01,
             3.62307594e+02,  7.78841376e+01,  4.31239967e+01,  1.25131822e+02,
             2.74219753e+01,  4.27845105e+01,  1.58849226e+02,  6.32157976e+01,
             7.78164639e+01,  2.72258932e+01,  7.40712277e+01,  4.02575948e+01,
             7.88091891e+01,  8.11972741e+01,  7.13543661e+01,  4.25904506e+01,
             6.89667672e+01,  6.34217343e+01])
```

Figura 101. SVM Radial modelo 26

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

RealValues	PredictedValues
68.0	63.20
0.0	66.85
87.0	162.34
87.0	135.10
0.0	31.80

Tabla 13. Predicciones Modelo 26

Modelo 27: Este modelo es del tipo SVM y usa un núcleo polinomial, probamos su desempeño en el dataset de pruebas, encontramos que el indicador RMSE es de 97.56.

```
[98] #Probamos usando un kernel Polynomial
     from sklearn.svm import SVR
     regressor = SVR(kernel='poly', C=5, gamma=0.05, degree=2)
     regressor.fit(X_train, y_train)

     SVR(C=5, cache_size=200, coef0=0.0, degree=2, epsilon=0.1, gamma=0.05,
         kernel='poly', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

```
[99] y_pred = regressor.predict(X_test)
     y_pred = sc_y.inverse_transform(y_pred)
     y_pred

     array([ 57.85989302,  52.9094781 , 114.93728149, 115.76019518,
             39.03842602,  67.28322999, 275.2021289 ,  62.35159359,
             51.86878798,  99.34950952,  54.17011772,  34.33657698,
             35.56985054,  88.08928956,  80.12206539,  58.12875178,
             44.73259309,  61.36128053,  47.70665547,  74.83492453,
             54.02243717,  32.56164832,  34.66324396,  65.25852436,
             45.65286924,  64.35859315,  46.64107692,  57.62412512,
             11.99619322,  -2.63134164, 640.59254921,  29.41546793,
             28.0771422 ,  21.67872954,  29.09168946,  72.39925858,
            356.46118931,  63.9814633 ,  56.90931061, 158.0922101 ,
            145.84001206,  66.93778674,  89.80739577,  66.92179101,
             65.17815551,  66.08939342,  68.71965144,  64.73559595,
             70.63230663,  71.86141486,  64.3045749 , 107.61722243,
             64.66789675,  64.24043725])
```

Figura 102. SVM Radial modelo 27

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

RealValues	PredictedValues
68.0	57.85
0.0	52.90
87.0	114.93
87.0	115.76
0.0	39.03

Tabla 14. Predicciones Modelo 27

Modelo 28: Este modelo es del tipo RN (Redes Neuronales) y se diseñó usando los siguientes parámetros:

- Capa de entrada: 111 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- 3capas ocultas: 50 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- Capa de Salida: 1 Neurona, Función de activación (Linear), Neurona de Sesgo (Verdadero).
- Batch Size: 32.
- Learning_rate=0.015
- Epocas: 300

El desempeño de este modelo es el siguiente:

Desempeño en entrenamiento:

- Coeficiente de determinación: 0.5677917657995519
- Coeficiente de correlación: 0.7535195855447634
- MSE: 51.24965158320232

Desempeño en Pruebas:

- Coeficiente de determinación: 0.5700773159796821
- Coeficiente de correlación: 0.7550346455492504
- MSE: 80.21064858167023

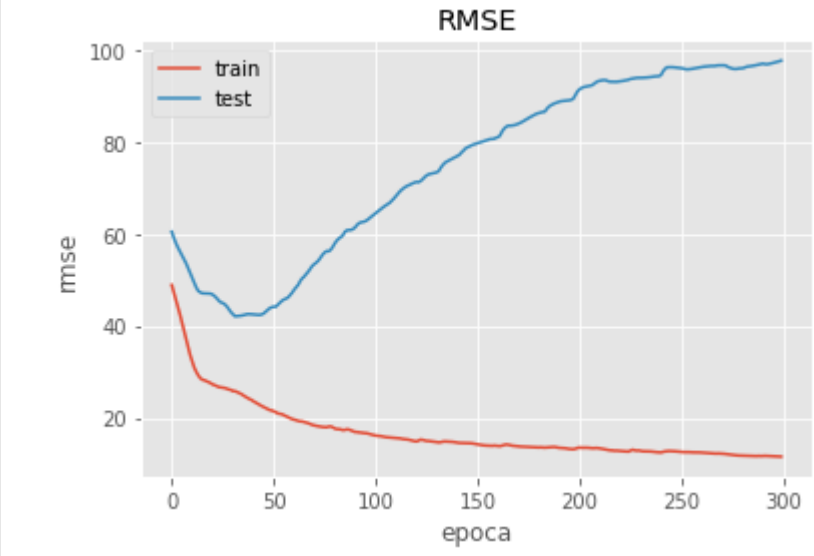


Figura 103. RN RMSE - modelo 28

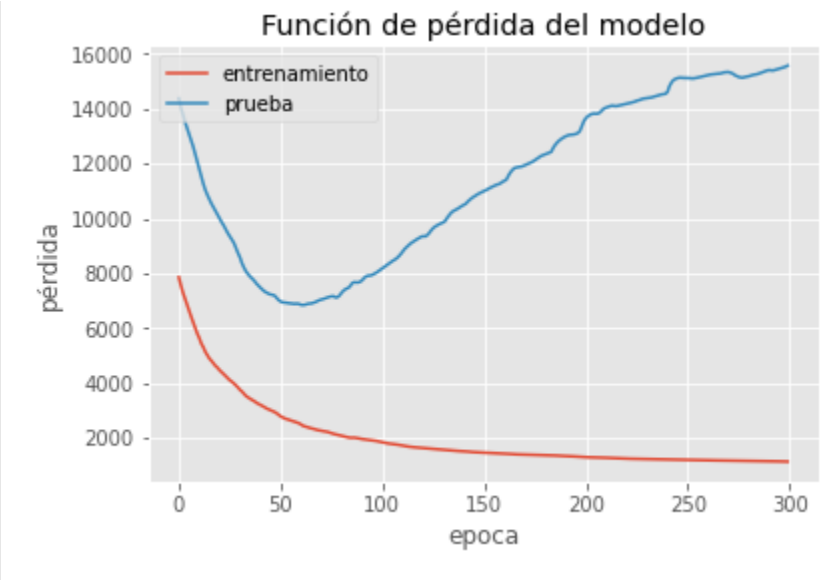


Figura 104. RN Función de Perdida - modelo 28

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

Valor Real	Valor Predicho
68.0	38.29
0.0	93.35
87.0	102.72
87.0	114.58
0.0	26.03

Tabla 15. Predicciones Modelo 28

Modelo 29*:* Este modelo es del tipo RN (Redes Neuronales) y se diseñó usando los siguientes parámetros:

- Capa de entrada: 111 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- 2 capas ocultas: 50 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- Capa de Salida: 1 Neurona, Función de activación (Linear), Neurona de Sesgo (Verdadero).
- Batch Size: 1.
- Learning_rate=0.015
- Epocas: 150

El desempeño de este modelo es el siguiente:

Desempeño en entrenamiento:

- Coeficiente de determinación: 0.9640652332633629

- Coeficiente de correlación: 0.9818682362024769
- MSE: 14.77752878457291

Desempeño en Pruebas:

- Coeficiente de determinación: 0.39746833637264656
- Coeficiente de correlación: 0.6304508992559583
- MSE: 94.95695620022266

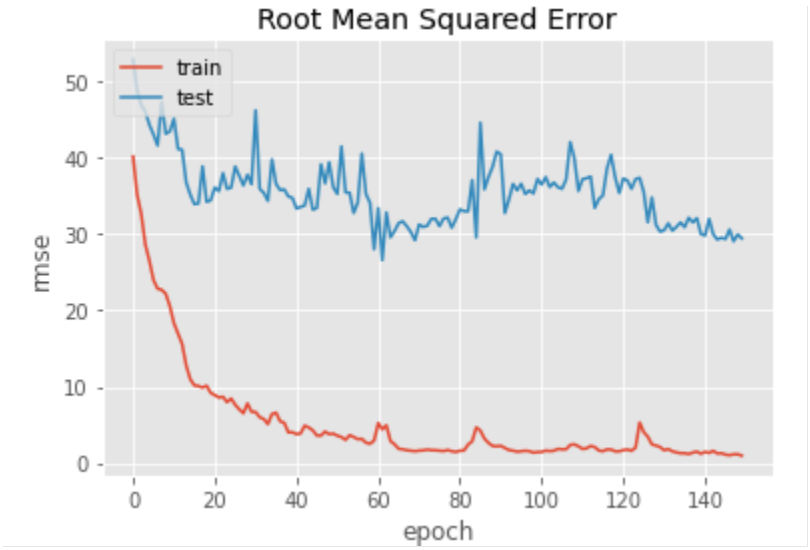


Figura 105. RN RMSE - modelo 29

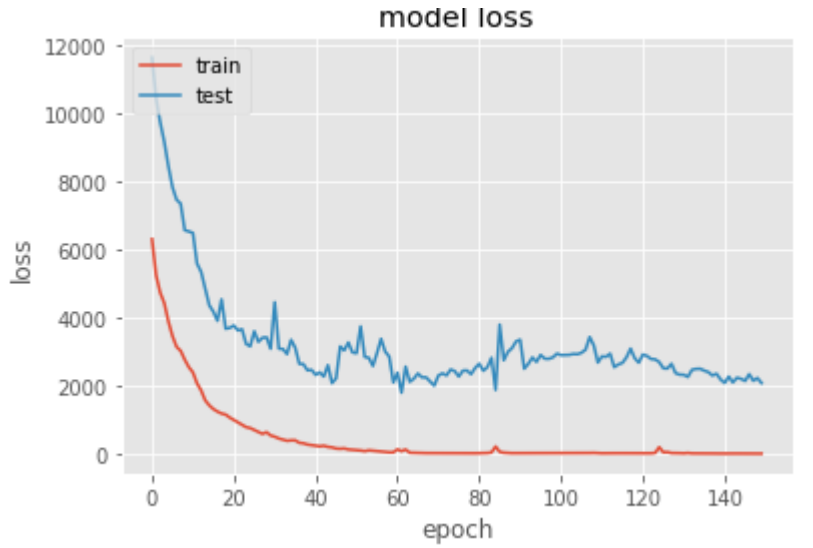


Figura 106. RN Función de Perdida - modelo 29

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

Valor Real	Valor Predicho
68.0	32.58
0.0	31.46
87.0	94.34
87.0	91.49
0.0	22.94

Tabla 16. Predicciones Modelo 29

Modelo 30: Este modelo es del tipo RN (Redes Neuronales) y se diseñó usando los siguientes parámetros:

- Capa de entrada: 111 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- 3capas ocultas: 50 Neuronas, Función de activación (Sigmoidal), Neurona de Sesgo (Verdadero).
- Capa de Salida: 1 Neurona, Función de activación (Linear), Neurona de Sesgo (Verdadero).
- Batch Size: 32.
- Learning_rate=0.015
- Epocas: 300

El desempeño de este modelo es el siguiente:

Desempeño en entrenamiento:

- Coeficiente de determinación: 0.858262678505969
- Coeficiente de correlación: 0.9264246750308246+0j
- MSE: 29.348540561370168

Desempeño en prueba:

- Coeficiente de determinación: 0.1525489076319536
- Coeficiente de correlación: 0.39057509858150663+0j
- MSE: 112.61446287434437

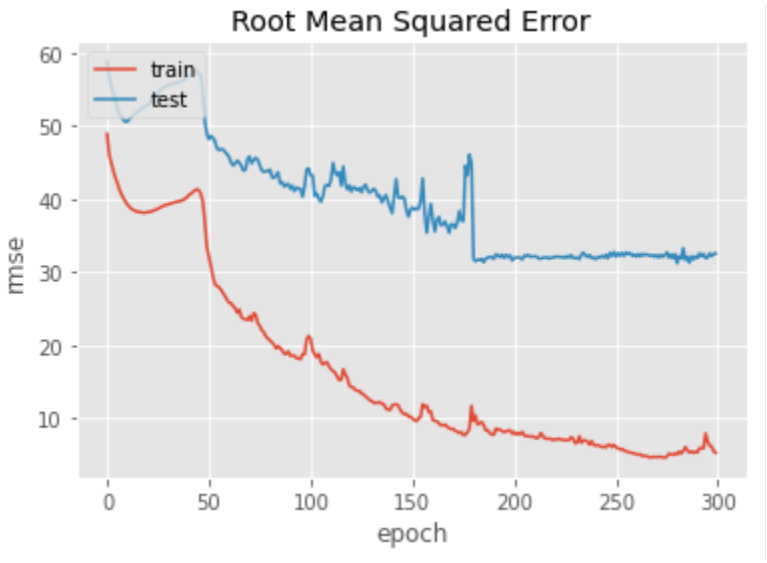


Figura 107. RN RMSE - modelo 30

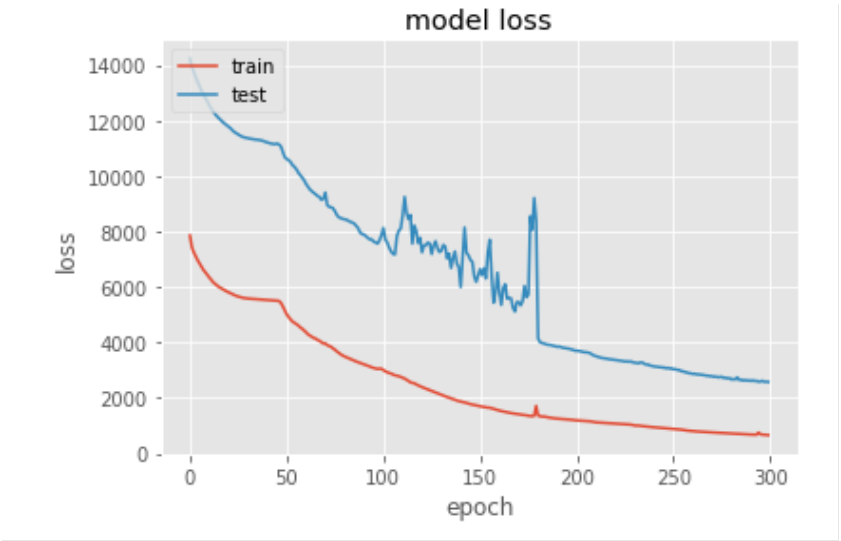


Figura 108. RN Función de Perdida - modelo 30

En la siguiente tabla se relacionan la predicción realizada por este modelo para los primeros cinco datos de prueba:

Valor Real	Valor Predicho
68.0	33.36
0.0	20.94
87.0	91.28
87.0	89.26
0.0	25.48

Tabla 17. Predicciones Modelo 30

Con el ánimo de validar si es posible obtener un mejor conjunto de parámetros para configurar los modelos de RN se realiza regularización, para ello se dispone del uso de las siguientes funciones:

- ***Sintonización del Batch Size y del Número de Epochs,*** Se configura un search grid con las siguientes constantes:

  batch_size = [10, 20, 40, 60, 80, 100]

  epochs = [50, 70, 100, 150, 170, 300]

- ***Sintonización del algoritmo de optimización,*** Se configura un search grid con las siguientes constantes:

  optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']

- ***Sintonización de la tasa de aprendizaje,*** Se configura un search grid con las siguientes constantes:

  learn_rate = [0.001, 0.01, 0.1, 0.2, 0.3]

- ***Sintonización del método de inicialización de pesos,*** Se configura un search grid con las siguientes constantes:

  init_mode = ['uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal', 'glorot_uniform', 'he_normal', 'he_uniform']

- ***Sintonización de la función de activación,*** Se configura un search grid con las siguientes constantes:

  activation = ['softmax', 'softplus', 'softsign', 'relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear']

El resultado del modelo después de aplicar las funciones de regularización usando varios Search Grid, es el siguiente:

Desempeño en entrenamiento:

- Coeficiente de determinación: -0.4059583061266383
- Coeficiente de correlación: 0.6371485746092809j
- MSE: 92.43376006633075

Desempeño en Pruebas:

- Coeficiente de determinación: -0.41430988608090513
- Coeficiente de correlación: 0.6436690811907195j
- MSE: 145.4819576442385

Como se puede observar el desempeño de este modelo no es superior a los demás modelos de RN planteados previamente.

5. Evaluación

A continuación, se relacionan los resultados bajo el indicador RMSE obtenidos al evaluar cada uno de los modelos diseñados:

#Modelo	*Tipo de Modelo*	*RMSE*	*Entrenamiento*	*Test*	*Predicción Modelo*
1	SVM Radial	7.92	1510, 1530, 1610, 1630, 1710, 1730, 1810, 1830 1910, 1930, 2010, 2030	2110	Número de inscritos por Clase
2	SVM Linear	6.92	1510, 1530, 1610, 1630, 1710, 1730, 1810, 1830 1910, 1930, 2010, 2030	2110	Número de inscritos por Clase
3	SVM Polinomial	8.06	1510, 1530, 1610, 1630, 1710, 1730, 1810, 1830 1910, 1930, 2010, 2030	2110	Número de inscritos por Clase
4	SVM Radial	3.79	1510, 1530, 1610	1630	Número de inscritos por Clase
5	SVM Linear	3.63	1510, 1530, 1610	1630	Número de inscritos por Clase
6	SVM Polinomial	4.06	1510, 1530, 1610	1630	Número de inscritos por Clase
7	SVM Radial	35.99	1510, 1530, 1610	1630	Número de inscritos por Asignatura
8	SMV Linear	29.16	1510, 1530, 1610	1630	Número de inscritos por Asignatura
9	SVM Polinomial	44.72	1510, 1530, 1610	1630	Número de inscritos por Asignatura
10	RN	60.89	1510, 1530, 1610	1630	Número de inscritos por Asignatura

11	RN	57.03	1510, 1530, 1610	1630	Número de inscritos por Asignatura
12	RN	27.01	1510, 1530, 1610	1630	Número de inscritos por Asignatura
13	SVM Radial	29.71	1630, 1710, 1730	1810	Número de inscritos por Asignatura
14	SVM Linear	27.31	1630, 1710, 1730	1810	Número de inscritos por Asignatura
15	SVM Polinomial	26.86	1630, 1710, 1730	1810	Número de inscritos por Asignatura
16	RN	45.86	1630, 1710, 1730	1810	Número de inscritos por Asignatura
17	RN	23.48	1630, 1710, 1730	1810	Número de inscritos por Asignatura
18	RN	27.84	1630, 1710, 1730	1810	Número de inscritos por Asignatura
19	SVM Radial	47.18	1810, 1830, 1910	1830	Número de inscritos por Asignatura
20	SVM Linear	23.86	1810, 1830, 1910	1830	Número de inscritos por Asignatura
21	SVM Polinomial	28.66	1810, 1830, 1910	1830	Número de inscritos por Asignatura
22	RN	51.81	1810, 1830, 1910	1830	Número de inscritos por Asignatura
23	RN	41.08	1810, 1830, 1910	1930	Número de inscritos por Asignatura
24	RN	49.72	1810, 1830, 1910	1930	Número de inscritos por Asignatura

25	SVM Radial	94.73	1930, 2010, 2030	2110	Número de inscritos por Asignatura
26	RN Linear	111.66	1930, 2010, 2030	2110	Número de inscritos por Asignatura
27	RN Polinomial	97.56	1930, 2010, 2030	2110	Número de inscritos por Asignatura
28	RN	80.21	1930, 2010, 2030	2110	Número de inscritos por Asignatura
29	RN	94.95	1930, 2010, 2030	2110	Número de inscritos por Asignatura
30	RN	112.65	1930, 2010, 2030	2110	Número de inscritos por Asignatura

Tabla 18. Resultados de los Modelos diseñados

En la siguiente tabla se relaciona el mejor modelo de acuerdo con el indicador RMSE por grupo de datos de entrenamiento y pruebas para SVM y RN.

#Modelo	*Tipo de Modelo*	*RMSE*	*Entrenamiento*	*Test*	*Predicción Modelo*
8	SVM Linear	29.16	1510, 1530, 1610	1630	Número de inscritos por Asignatura
12	RN	27.01	1510, 1530, 1610	1630	Número de inscritos por Asignatura
15	SVM Polinomial	26.86	1630, 1710, 1730	1810	Número de inscritos por Asignatura
17	RN	23.48	1630, 1710, 1730	1810	Número de inscritos por Asignatura
20	SVM Linear	23.86	1810, 1830, 1910	1930	Número de inscritos por Asignatura
23	RN	41.08	1810, 1830, 1910	1930	Número de inscritos por Asignatura

Tabla 19. Resultado de los modelos diseñados con el mejor índice RMSE

Es importante señalar que se usaron parámetros diferentes en el diseño de los 6 modelos base (3 SVM y 3 RN), posteriormente estos modelos se probaron en cada uno de los conjuntos de datos de entrenamiento y pruebas. A continuación, se describen los parámetros utilizados en el diseño base y el resultado promedio por conjunto de datos de entrenamiento:

- Modelos SVM:

Kernel	*C*	*Degree*	*Epsilon*	*Gamma*	*Max_iter*	*Promedio RMSE*
(Radial)	1	3	0.1	Scale	-1	37,66
(Linear)	1	3	0.1	Scale	-1	26,66
(Polinomial)	5	2	0.1	0.05	-1	33,66

Tabla 20. Resultados promedio de los Modelos diseñados SVM

- Modelos RN:

Capa de Entrada	*Capas intermedias*	*Épocas*	*Batch Size*	*Activación*	*Promedio RMSE*
111 neuronas	1 (50 neuronas)	300	32	Sigmoid	53
111 neuronas	2(50 neuronas)	150	1	Sigmoid	40,33
111 neuronas	3(50 neuronas)	300	32	Sigmoid	35

Tabla 21. Resultados promedios de los Modelos diseñados RN

Como se puede observar en las tablas previas el mejor rendimiento lo presentan los modelos SVM (Support Vector Machine), en particular el diseñado con un núcleo linear, su promedio RMSE para los conjuntos de entrenamiento y pruebas es de *26.66.*

6. Despliegue

6.1. Planear Despliegue.

Para poder desplegar este proyecto como un negocio real sería necesario en primer lugar tener acceso a las fuentes de datos del negocio de forma directa, es decir las fuentes en primera línea que contiene toda la información relativa a inscripciones, notas y retiros durante todos los ciclos del Programa de Ingeniería de Sistemas. A partir de ahí, los pasos a seguir serían los mismos que se han seguido en este documento desde la comprensión del negocio hasta la implementación. Si bien, cabe decir que habrá algunas fases adicionales, como la integración de las fuentes de información que se encuentran en la plataforma PeopleSoft, que es una de las herramientas que contiene toda la información de los estudiantes y su comportamiento en la universidad, también la de comprensión y preparación de los datos, que en el negocio real probablemente sean más complejas y tengan cambios importantes en el tiempo, por lo tanto, llevarán más tiempo que en este proyecto ya que se puede esperar que las fuentes de información pueda contener muchos más registros y estos mismos contengan más ruido que las fuentes de información suministradas, limitadas, filtradas y creada específicamente para el uso de este proyecto.

En segundo lugar, sería necesario que en el negocio (la Universidad) implemente una base de datos donde se almacene la información que se logre extraer de las diferentes fuentes de información o aplicaciones, de no ser así se tendrían dos opciones, la primera sería exportar archivos de las diferentes fuentes de información como lo tenemos actualmente en este proyecto y la segunda opción sería utilizar otro software de minería de datos distinto al utilizado en este proyecto, para la unificación de estas fuentes de información.

Para este propósito se podría utilizar alguna de las herramientas que mejor se adapte a la integración de fuentes de datos original de acuerdo con los cambios o actualizaciones tecnológicos que se estén implementado en el área de negocio y el área de TI actualmente, para esto sería necesario hacer un estudio previo que determine que herramienta es la más apropiada de acuerdo con la arquitectura de negocio actual con el fin de determinar dicha viabilidad.

6.2.Planear la Monitorización y Mantenimiento.

La supervisión y mantenimiento de la implementación del presente proyecto es una fase importante del mismo, debido a que los datos que se procesan con mucha frecuencia pueden ser modificados por el personal de la universidad. Los datos pueden ser modificados por diferentes motivos como haber realizado una codificación incorrecta, haber dejado información inconclusa sin ningún tipo de definición, las fuentes de información son diferentes en nombres o estructura etc.

El volumen de estos datos en movimiento es grande, motivo por el cual la extracción de las muestras debe ser realizada cuidadosamente y realizando siempre copias de seguridad de los datos explotados en cada proceso. La minería de datos debería ser realizada en periodos de cuatro meses (cuatrimestres) ya que esta es la medida de tiempo utilizada en la universidad para realizar los exámenes y asignar las notas finales a los alumnos, sin embargo, esta medida podría variar en cualquier momento en función del plan de estudios que esté vigente.

Como plan de supervisión y mantenimiento se podría establecer los siguientes procesos:

- Extracción y almacenamiento cuatrimestral de los datos guardando la información obtenida en formato de hoja de cálculo (.xslx).
- Los archivos de la explotación de datos deberán ser guardados como soporte en la nube de la propia universidad, almacenándolos por ejemplo en carpetas ordenadas por procesos cuatrimestrales.
- Los resultados obtenidos en cada explotación de datos deberán ser llevados a formato de hoja de cálculo y generar gráficas de distintos tipos para una mejor visualización e interpretación de los resultados obtenidos en cada ciclo.

6.3.Producir el Informe Final.

En este paso se está presentado en un informe final resumiendo los puntos importantes del proyecto y la experiencia adquirida durante su desarrollo. El público al que va dirigido este informe es el Programa de Ingeniería de Sistema de la universidad que a su vez solicita la programación de clases a los departamentos correspondientes, de tal manera que se pueda estudiar la situación actual y tomar decisiones acertadas, planes correctivos o de contingencia para inscripciones de estudiantes. Cabe decir que parte de este informe final será presentado de manera oral con una presentación, por lo que en este apartado solamente haremos un breve resumen.

El uso de la metodología CRISP-DM en este proyecto ha permitido encontrar un comportamiento predictivo a la hora de predecir cuantitativamente las inscripciones de asignaturas de los estudiantes por ciclo académico, también se logró brindar como valor adicional información del comportamiento y estado de deserción de los estudiantes en cada una de las asignaturas durante cada ciclo académico esto para comprender la situación de los estudiantes. Se ha podido encontrar un plan de extracción, normalización, y codificación de datos para la realización de procesos de minería de datos cuatrimestrales que equivalen a un ciclo académico.

De los tres objetivos de minería de datos iniciales que se habían fijado, se cumplieron todos en su totalidad (objetivos 1, 2 y 3). Además, al margen de estos objetivos, se han sacado otras conclusiones a partir de los datos estudiados, concretamente se han identificado las asignaturas con más inscripciones por los estudiantes, motivos de retiro de asignaturas, cantidad de estudiantes con planes de becas, capacidad de atención por docentes.

Repasando las diferentes etapas que hemos seguido para llegar al objetivo:

La primera etapa ha sido la más laboriosa por no tener comprensión de los procesos ejecutados por la universidad para la inscripción de asignaturas para un nuevo ciclo, esto implico acordar varias sesiones con el analista encargado de la universidad para validación y conocimiento de los procesos de negocio, y esto estaba enmarcado bajo criterios de disponibilidad y tiempos limitados de su parte, sin embargo, siempre existió la disposición para atendernos, en estas sesiones se acordaron validaciones de procesos, consultas de fuentes de datos, avances de predicción en cuanto a resultados y validaciones de tableros de control, otra parte de esta fase que influyo fue no tener accesos a las fuentes de datos directas por políticas y seguridad de la universidad, la información entregada no contiene datos sensibles que puedan afectar a los estudiantes o docentes sin embargo al realizar la exploración encontramos que no es suficiente información para predecir con márgenes de error de un 90%, se realizó un análisis de la estructura de los datos y la información contenida.

Se realizaron consultas y limpieza significativas de los datos en Python para tener muestras representativas, y sacar más conclusiones al margen de los objetivos iniciales de la minería. A continuación, se realizó la elección de las técnicas de modelado y la ejecución de dichas técnicas sobre los datos empleando la herramienta escogida para ello (Python). Esta herramienta facilitó por completo la aplicación de los modelos ya que nos permitieron ver de manera muy intuitiva y visual cuales eran las técnicas más adecuadas para las fuentes de datos trabajadas. Una vez obtenidos los modelos, se analizaron para determinar la adecuación o no de los mismos. En este caso determinamos un modelo predictivo más eficiente que podrían ser válidos para nuestros objetivos. Por último, se realizó la creación de tableros de control que permitiera de forma agradable entender a los interesados la exploración de las fuentes de información y la perdición ejecutada, esta validación se realizó en conjunto con el analista encargado del programa y se obtuvo retroalimentación de su visión asertiva como negocio.

7. Conclusiones

Durante el desarrollo del proyecto se evidenció la importancia de entender el negocio además de sus requisitos particulares. El esquema de arquitectura por procesos permitió identificar las oportunidades de mejora dentro del mismo, además de los actores, datos e información involucrada en la programación de clases para un periodo académico especifico dentro del departamento de ingeniería de Sistemas para el grado académico de pregrado.

Es necesario entender que los datos obtenidos de la identificación de los procesos involucrados en la programación de clases requieren de una trasformación previa antes de ser usados en el diseño de un modelo analítico, por lo tanto, se investigaron técnicas y herramientas para transformar estos datos; siendo el lenguaje de programación Python y algunas de sus librerías especializadas en la gestión de información las opciones seleccionadas. Una vez identificados los datos base se aplicaron sobre ellos herramientas de exploración limpieza e imputación de datos como Pandas y Skilearn. Es importante señalar que se dispuso de la creación de funciones específicas debido a los requerimientos puntuales planteados en los objetivos.

Dentro del proceso de investigación del estado del arte se identificaron soluciones que tenían componentes en común al problema planteado, se usaron y se modificaron algunas de estas características para adaptarlas a la solución final. En las fases iniciales de desarrollo se plantearon entrevistas que permitieron obtener información relacionada con la programación de clases y que permitió principalmente diseñar un diagrama de grafos dirigido para esquematizar el plan de estudios del programa de ingeniería de sistemas, a través del uso de este diagrama se identificaron variables significativas para el desarrollo del modelo analítico, como por ejemplo los nodos (asignaturas) que son más importantes que otros, debido a que se consideran “cuellos de botella”.

Posterior a la creación de la vista minable sobre la cual realizar el diseño del modelo analítico, se decidió investigar acerca de las opciones disponibles para atender de forma más eficiente la predicción de la variable objetivo “Número de estudiantes inscritos para un periodo académico especifico.”, se dispuso del uso de las técnicas de SVM (Support Vector Machine) y Redes Neuronales para diseñar este modelo.

El conjunto de datos para entrenamiento y pruebas se dividió en periodos académicos específicos, es importante señalar que se usaron tres periodos para entrenamiento y uno para pruebas y así hasta formar 4 conjuntos de datos para experimentación, vale la pena resaltar que los tres primeros conjuntos corresponden a periodos de tiempo regulares en donde no se había presentado la emergencia social y sanitaria por Covid-19, por tanto los modelos entrenados usando periodos de tiempo posteriores al 2019 no son tan eficientes para predecir el número de estudiantes inscritos como los que fueron entrenados con datos previos a este periodo de tiempo.

Los modelos que tienen mejor rendimiento son los que fueron desarrollados usando SVM, claro está para periodos anteriores a la emergencia social y sanitaria por Covid-19, para el periodo de tiempo comprendido entre el año 2020 y el año 2021, el modelo que presenta mejor rendimiento es el que fue desarrollado usando redes Neuronales, es importante señalar que el error cuadrático medio es superior a cualquiera de los modelos SVM desarrollados para periodos académicos previos.

De los modelos que fueron desarrollados usando SVM el de mejor eficiencia promedio para los 3 conjuntos de datos iniciales, fue el que se diseñó usando un núcleo linear, su RMSE promedio es de 26.66.

De los modelos que fueron desarrollados usando RN el de mejor eficiencia promedio para los 3 conjuntos de datos iniciales, fue el que se diseñó usando 3 capas intermedias cada una con 50 neuronas y 300 épocas y un bacht size de 32, su RMSE promedio es de 35.

Por tanto, para periodos de tiempo regulares se sugiere usar el modelo SVM diseñado usando un núcleo Linear, debido a que su rendimiento promedio fue superior al de los demás modelos.