

**ANÁLISIS DE DATOS**

**Portafolio de evidencias**

**Estudiantes:**

**Catalina Chavarría Madriz**

**Profesor:**

**JOSUE ZUNIGA HERNANDEZ**

**Fecha: 15 Agosto 2023**

**Índice**

[**Semana 1 – 16 de mayo** 3](#_Toc143024919)

[**Semana 2 – 23 de mayo** 4](#_Toc143024920)

[**Semana 3 – 30 de mayo** 5](#_Toc143024921)

[**Semana 4 – 6 de junio** 8](#_Toc143024922)

[**Semana 5 – 13 de junio** 10](#_Toc143024923)

[**Semana 6 – 20 junio** 13](#_Toc143024924)

[**Semana 7 – 27 de junio** 17](#_Toc143024925)

[**Semana 8 – 4 julio** 17](#_Toc143024926)

[**Semana 9 – 11 julio** 18](#_Toc143024927)

[**Semana 10 – 25 julio** 24](#_Toc143024928)

[**Semana 12 – 01 de agosto** 34](#_Toc143024929)

[Bibliografía 38](#_Toc143024930)

# **Semana 1 – 16 de mayo**

1. Síntesis de los contenidos abordados en la sesión:

Para esta primera clase fue introducción de cómo se llevaría a cabo el curso, que haríamos y que veríamos.

También el profesor nos explicó como es proyecto final y lo que se debe de hacer.

Adicional pudimos ver los entornos donde estaríamos trabajando, como descargarlos e iniciarlos, como Anaconda y GitHub.

1. Desarrollo de actividades y lecturas complementarias:

Para esta primera clase sobre los conceptos de minería y analítica de datos.

<https://learn.microsoft.com/es-es/analysis-services/data-mining/data-mining-concepts?redirectedfrom=MSDN&view=asallproducts-allversions&viewFallbackFrom=sql-server-ver16>

1. Evidencias:

Adjunto repositorio con la lectura dada en esta primera sesión <https://github.com/CataCh1505/Primera-lectura>

1. Lecciones Aprendidas:

Aprendí como ajustar y crear cuenta en Git Hub y conocimos un poco del entorno donde se estará trabajando el resto de la clase.

# **Semana 2 – 23 de mayo**

1. Síntesis de los contenidos abordados en la sesión:

Hablamos sobre las metodologías de CRISP – DM y BADIR:

* CRISP – DM

Esta metodología aprendí que es importante porque nos da una guía para llevar a cabo proyectos de minería de datos de forma exitosa. Nos ayuda a evitar errores comunes, saber cómo gestionar riesgos y asegura que los resultados sean útiles para en la minería de datos.

* BADIR

El marco BADIR es un modelo de toma de decisiones que ayuda a las empresas a tomar decisiones eficaces utilizando datos.

1. Desarrollo de actividades y lecturas complementarias

Con base a estas lecturas dadas, se crearon grupos para leer y realizar una exposición sobre dicha lectura.

Para el grupo en el que me encontré, hablamos sobre CRISP - DM.

1. Evidencias:

Adjunto repositorio con la lectura CRISP-DM con la exposición

<https://github.com/CataCh1505/Segunda-lectura>

1. Lecciones Aprendidas

De estas primeras lecturas he aprendido que estas metodologías son importantes porque nos proporcionan una guía para realizar tareas y la resolución de problemas. Son herramientas valiosas para lograr objetivos de manera efectiva y con resultados predecibles.

Nos permite identificar y priorizar las tareas más importantes, principalmente agilizar procesos de desarrollo porque podemos usar herramientas.

# **Semana 3 – 30 de mayo**

1. Síntesis de los contenidos abordados en la sesión:

Para esta clase primero vimos sobre HTML en Jupyter.

Este lenguaje de marcado de hipertexto se utiliza mayormente para la creación de páginas web y aplicaciones. Así se ve la estructura en HTML.

<**html**> - Abre (inicio)

<**head**> - Encabezado (inicio)

<**title**> Mi primer página web </**title**> - Título de la pestaña

</**head**> Cierra (fin)

<**body**> Cuerpo. Parte de programación de html

<**h1** **color**=**red**> mi nombre es: Catalina </**h1**> - esto es un título

<**img** **src**="**image/profile.jpg**" **width**="**300**" **height**="**300**"> - agrega imagen

<**h2**> Curso Análisis de datos</**h2**> - esto es un título

<**h3**>  Universidad Castro Carazo </**h3**> - esto es un título

<**a** **href**="**https://www.crhoy.com**"> PERIÓDICO LOCAL </a> - agrega una vinculo, ya sea video o artículo de internet.

</**body**> Cierra (fin)

</**html**> se cierra (fin)

1. Desarrollo de actividades y lecturas complementarias

Para esta clase junto al profesor empezamos a utilizar el entorno de Jupyter a familiarizarnos con ella, como en este ejemplo de cómo es y cómo funciona. Para luego empezar a utilizar HTML en Jupyter.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Evidencias:

Para esta clase empezamos a poner en práctica los elementos de una estructura en HTML como agregar títulos, subtítulos y como insertar imágenes en el entorno Jupyter, como en las siguientes screenshots.

Imagen de la pantalla de un celular con la imagen de una flor amarilla

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

También empezamos a construir el proyecto que se usará en todo el curso.



1. Lecciones Aprendidas

El lenguaje HTML define la estructura y el contenido de las páginas. Es esencial para la construcción y el funcionamiento de la web tal como la conocemos.

# **Semana 4 – 6 de junio**

Esta semana no hubo clases, pero el profesor nos asignó lecturas para la siguiente semana.

Python para todos: 5 formas de cargar datos para tus proyectos de Machine Learning

* Cargar los ficheros CSV usando el módulo CSV de la librería estándar de Python: Para extraer los datos de un fichero csv, primero se abre con la función **open**(). Después la función **csv.reader**(), lee línea a línea el fichero, y hace una lista de todas las columnas en el objeto reader.
* Cargar los ficheros CSV usando NumPy: usamos la función **numpy.loadtxt**() de la librería NumPy. Estas funciones permiten cargar datos csv en forma de matrices, lo que facilita el análisis y el procesamiento de los datos.

“Esta función asume que el fichero no tiene cabeceras, y que todos los datos tienen el mismo formato. Como no es el caso de nuestro dataset sobre diabetes, usaremos el modificador "skiprow" para indicar que no debe tener en cuenta la primera fila.” Cita textual: (Python para todos: 5 formas de cargar datos para tus proyectos de Machine Learning, 2022)

* Cargar los ficheros CSV usando Pandas: para cargar archivos csv se utiliza la biblioteca de Pandas la función **pandas.read\_csv**().

“Los dataframes de Pandas son unas estructuras de datos diseñadas especialmente para facilitar el trabajo del analista y científico de datos. Permiten trabajar con datos de todo tipo (enteros, decimales, cadenas de texto) dispuestos en forma de tablas, incluso con series temporales.” Cita textual: (Python para todos: 5 formas de cargar datos para tus proyectos de Machine Learning, 2022).

* Cargar los ficheros desde la URL: se agrega la URL del archivo CSV en lugar de un nombre de archivo local. Pandas descargará automáticamente el archivo CSV desde la URL y lo cargará en un dataFrame.
* Cargar ficheros de pruebas desde librerías (Staatsmodels, etc): Statsmodels es una biblioteca para modelos estadísticos. Para poder cargar datos se utiliza la función como **get\_rdataset()**.

Importar datos en Python de diferentes formas

Hay una gran variedad de archivos que se pueden usar como fuente de datos:

* Archivos planos – csv, txt, tsv, etc: Las bibliotecas como Pandas y Numpy son populares para el análisis de datos debido a su riqueza de funciones y facilidad de uso.

Para Numpy se usa **numpy.recfromcsv** para cargar datos ASCII almacenados en un archivo separado por comas.

Para Pandas es fácil se usa **pd.read\_csv** y listo. También admite opcionalmente iterar o dividir el archivo en fragmentos.

* Archivos encurtidos: conocidos como archivos pickle (.pkl), es una forma beneficiosa de guardar y cargar objetos en Python.

“Se usa para serializar y deserializar una estructura de objeto de Python. Cualquier objeto en Python se puede escanear para que pueda guardarse en el disco.” Cita textual: (Importar datos en Python de diferentes formas, 2018)

* Excel spreadsheets: Para importar datos en Python desde Excel spreadsheets se utiliza la función **pandas.read\_excel()** de la biblioteca pandas.
* SAS y archivos Stata:

SAS: es un software para análisis y gestión de datos. df\_sas = file.to\_data\_frame() para abrir archivos SAS

Stata: es un software estadístico para el análisis, modelado estadístico y generación de informes. df\_sas = file.to\_data\_frame() para abrir archivos Stata.

Se utilizar pandas para leer datos desde archivos SAS y Stata para trabajar con los datos de una manera eficiente.

* MATLAB:

MATLAB: es un entorno de programación que se usa para cálculos numéricos.

Para escribir y leer archivos MATLAB se usa el paquete scipy.io **mat = scipy.io.loadmat('some\_project.mat')**

* API: se puede utilizar bibliotecas como http.client para realizar solicitudes HTTP y obtener los datos.

**import requests**

**r = requests.get('https://www.example.com/some\_endpoint')**

# **Semana 5 – 13 de junio**

1. Síntesis de los contenidos abordados en la sesión:

Para esta semana empezamos a trabajar en 3 puntos:

* Importación de librerías: que se utilizaran para este proyecto que son pandas, numpy y matplotlib.
* Carga de datos: de los dos archivos csv Autos y autos que debe de estar en la carpeta del proyecto. Y se cargan los dataframes para los dos autos y los nombres de las columnas

Tenemos varias funciones como:

* Df\_nombredataframe.Shape: para saber si los archivos se cargaron y cuantas variables tiene.
* Head(): muestra las primeras filas de un DataFrame.
* Tail(): se utiliza para mostrar las últimas filas de un DataFrame
* Revisión de encabezados: Lo que se busca es estandarizar el nombre de cada columna de los sets de datos que se carguen a los proyectos de Python. En este apartado se utiliza la función columns para identificar cuales tenemos en el dataframe y luego rename para cambiar el nombre de la columna existente.
* Integrar set de datos: esto es para cuando queremos crear tuplas. En esta tupla tiene los archivos que se cargan al proyecto. En una tupla mete dos informaciones y se crea un nuevo dataframe que concatene lo que tiene la tupla.

1. Desarrollo de actividades y lecturas complementarias

Desarrollamos en clase los puntos anteriormente vistos importar librerías, cargar nuestros datos e integrar datos. Para esta clase se debió haber leído las lecturas de la semana pasada.

1. Evidencias

Imagen que contiene Aplicación

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Lecciones Aprendidas

Con el complemento de las lecturas de la semana pasada, se aprendió como importar las librerías y cuales son las que se usaran en el curso. Se aprendió como se debe de cargar correctamente los datos para usar en nuestro dataframe y como integrar tuplas en el proyecto.

# **Semana 6 – 20 junio**

1. Síntesis de los contenidos abordados en la sesión:

Para esta clase empezamos con el punto 4 que son manejar valores faltantes:

* Identificar valores faltantes: esto se usa el **replace**: remplaza todos los signos de pregunta por NAN.
* Contar cantidad de valores faltantes: se crea un dataframe temporal, que es igual al datafram de Autos, el isnull busca en dataframe Autos donde hay campo nulo le pone True y donde no esta es False.
* Imputación de valores numéricos: se crea variable, se va al dataframe Autos y convertir esa columna a tipo dato numerico y calcula la media de los valores.
* Imputación de valores categóricos: se utiliza la función **value\_counts().idxmax()** esto busca el valor con mayor frecuencia en la columna doors. Otra función que tenemos es **replace(np.nan, 'four', inplace = True)** donde reemplaza todo los nan por la palabra 'four'.
* Eliminar filas: si tiene columna con muchos valores faltantes se pueden eliminar la fila, para esto utilizamos **dropna** se utiliza para eliminar las filas o columnas que contienen valores faltantes (NaN) de un DataFrame.

1. Desarrollo de actividades y lecturas complementarias

Practicamos estos nuevos conceptos en nuestros proyectos.

1. Evidencias

Aplicación

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Imagen que contiene Calendario

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Imagen que contiene Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. Lecciones Aprendidas

Para algo importante de como eliminar filas que tenemos en nuestro dataframe tenemos varias formas de como poder eliminar lo que no necesitamos y aprendemos como identificar valores faltantes de nuestro dataframe.

# **Semana 7 – 27 de junio**

Para esta sesión no hubo clases.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

# **Semana 8 – 4 julio**

Esta sesión fue corta, para poder seguir trabajando en el proyecto.



# **Semana 9 – 11 julio**

1. Síntesis de los contenidos abordados en la sesión:

Para esta clase aprendimos sobre el punto 5 estandarizar los datos en nuestro dataframe:

* Verificar dominio de variables: en este apartado podemos convertir las columnas a variables, como int o float dependiendo de lo que contenga nuestros datos. Esto genera que deja de ser una variable object a int.
* Convertir valores numéricos en función de la interpretación del medio: se debe ver la distribución de datos, se usa la campana de Gaus.

Aquí le decimos a dfAutos que cree una nueva columna y esta contiene una formula 235/dfAutos de los campos en ciudad.

* Normalizar valores entre 0 y 1: creamos dataframe temporal que es igual lo que tiene dfAutos en las columnas nuevas que se crearon para poder hacer comparaciones. También podemos visualizar compraciones con la función max() utilizando una formula: dfAutos["city L/100km (norm)"] = dfAutos['city L/100km (Calc)']/dfAutos['city L/100km (Calc)'].max()
* Creación de rangos de valores (Binning): al analizar los datos, la necesida de crear bings para poder resumir datos y visualizarlos de la mejor forma. La que aprendimos en clase fue:

bins = np.linspace(48,262,4)

bins

1. Desarrollo de actividades y lecturas complementarias

Realizamos lectura sobre **Matplotlib Functions** para la siguiente semana.

**Data Visualization Libraries** donde nos habland de librerías de visualización de datos, seaborn, pandas, etc.

Todas estas lecturas para familiarizarnos con la creación de gráficos para la siguiente semana.

1. Evidencias

Texto

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto, Aplicación

Descripción generada automáticamente

1. Lecciones Aprendidas

Para este punto empezamos a estandarizar los datos de nuestro dataframe y poder realizar función de contabilidad para visualizar datos que requerimos.

# **Semana 10 – 25 julio**

1. Síntesis de los contenidos abordados en la sesión:

Para estas ultimas clases vemos sobre el ultimo de tema de graficar los datos.

Para crear gráficos debemos importar las librerías

*import matplotlib.pyplot as plt #Para generar las graficas*

*import matplotlib as mpl #Para configurar las gráficas.*

**mpl.rcParams["axes.spines.bottom"] =True**, se utiliza para establecer y configurar parámetros globales que afectan el comportamiento de los gráficos que se generan con la librería de Matplotlib.

*gf = dfAutos.groupby(['make']).count()*

*gf = gf.iloc[:,:1]*

*gf = gf.sort\_values('symboling', ascending = False)*

*gf*

La primera línea de código es para agrupar el DataFrame Autos por la columna **'make'**. Luego, usamos **count()** para contar el número de modelos por cada fabricante.

La segunda línea **iloc()** se utiliza para acceder a filas y columnas de un DataFrame en función de sus índices. En este caso selecciona todas las filas

La tercera nos hace ordenar de manera ascendente de la columna symboling los datos.

Vimos la estructura de cómo crear gráficos

Gráfico

Descripción generada automáticamente con confianza baja

La línea de código de arriba está creando una figura (fig) y un conjunto de ejes (ax) utilizando la biblioteca Matplotlib.

**figsize(15, 6.9)**: Define el tamaño de la figura en pulgadas.

**dpi=100**: Establece la resolución en puntos por pulgada.

**Bar\_colors** establecemos un conjunto de colores que se utilizarán en las barras en el gráfico.

**ax.barh**: crea el gráfico de barras horizontales utilizando los datos y las etiquetas.

plt.show(): Finalmente, esta línea muestra el gráfico en la figura que creaste

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**title = "Grafico 1: Vehiculos"** se pone el título a la figura.

**plt.suptitle(title, ...):** se utiliza para agregar un título superior a la figura. Adentro van los argumentos que aparecerán en el gráfico.

**horizontalalignment='left':** establece la alineación horizontal del título a la izquierda.

**x=0.001, y=0.87:** estos valores especifican las coordenadas x e y de la esquina superior izquierda del título en la figura.

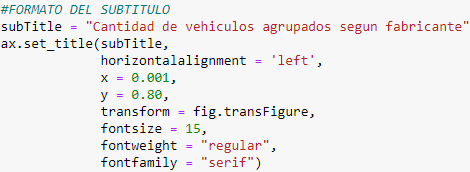
**transform=fig.transFigure**: establece cómo se interpretan las coordenadas x e y.

**fontsize=20**: define el tamaño de fuente del título.

**fontweight="bold"**: establece el grosor de la fuente en negrita.

**fontfamily="serif"**:eEspecifica la familia de fuentes.

**fontname="Arial"**: define el nombre de la fuente como "Arial".



**subTitle = "Cantidad de vehiculos agrupados segun fabricante"**: se pone el subtítulo al conjunto de ejes.

**ax.set\_title(subTitle, ...)**: establece el subtítulo en el conjunto de ejes. Adentro van los argumentos que aparecerán en el gráfico.

**horizontalalignment='left'**: establece la alineación horizontal del subtítulo a la izquierda.

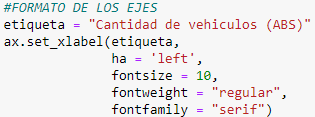
**x=0.001, y=0.80**: especifican las coordenadas x e y de la esquina superior izquierda del subtítulo en el conjunto de ejes.

**transform=fig.transFigure**: establece cómo se interpretan las coordenadas x e y.

**fontsize=15**: define el tamaño de fuente del subtítulo.

**fontweight="regular"**: establece el grosor de la fuente como regular.

**fontfamily="serif"**: especifica la familia de fuentes como "serif".



**etiqueta = "Cantidad de vehiculos (ABS)"**: se pone la etiqueta para el eje x.

**ax.set\_xlabel(etiqueta, ...)**: establece la etiqueta en el eje x del conjunto de ejes. Adentro van los argumentos que aparecerán en el gráfico.

**ha='left'**: establece la alineación horizontal de la etiqueta en el eje x a la izquierda.

**fontsize=10**: define el tamaño de fuente de la etiqueta.

**fontweight="regular"**: establece el grosor de la fuente como regular.

**fontfamily="serif"**: Especifica la familia de fuentes como "serif".

CONFIGURACION DE COORDENADAS DE LA ETIQUETA DEL EJE X



**x=0**: establece la coordenada x de la etiqueta del eje x en 0.

**y=-0.07**: establece la coordenada y de la etiqueta del eje x en 0.07.

**transform=ax.transAxes**: especifica que las coordenadas x e y son relativas a las coordenadas del conjunto de ejes ax en lugar de las coordenadas del gráfico.

CONFIGURACION EL TAMANO DEL TEXTO DEL EJE Y



**axis='both'**: especifica los parámetros en ambos ejes

**labelsize=10**: establece el tamaño fuente de las etiquetas en 10.

CONFIGURACION LA UBICACION DEL GRAFICO



**plt.subplots\_adjust(top=0.80)**: ajustar los márgenes de la figura

**plt.show()**:muestra el gráfico.

Gráfico, Histograma

Descripción generada automáticamente

1. Desarrollo de actividades y lecturas complementarias

Adjunto repositorio con la presentación de la semana pasada.

<https://github.com/CataCh1505/Librerias.git>

1. Evidencias

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

Imagen que contiene Texto

Descripción generada automáticamente

Gráfico

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media

Gráfico

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Gráfico

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente con confianza baja

Gráfico, Gráfico circular

Descripción generada automáticamente

1. Lecciones Aprendidas

Aprendimos como graficar los datos de nuestro dataframe, y aprendimos distintos tipos de gráficos.

También

# **Semana 12 – 01 de agosto**

1. Síntesis de los contenidos abordados en la sesión:

En esta ultima sesión, seguimos viendo de los gráficos y practicando con preguntas para poder realizar grafico de lo que se pedía.

1. Desarrollo de actividades y lecturas complementarias
2. Evidencias

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media

Gráfico

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Imagen que contiene Texto

Descripción generada automáticamente

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

1. Lecciones Aprendidas

Para esta última sesión seguimos practicando gráficos y realizando ejercicios con el profesor.

# Bibliografía

*Importar datos en Python de diferentes formas*. (2018). Obtenido de https://sitiobigdata.com/2018/10/01/diferentes-formas-importar-datos-python/#

*Python para todos: 5 formas de cargar datos para tus proyectos de Machine Learning*. (2022). Obtenido de https://telefonicatech.com/blog/python-5-formas-de-cargar-datos-csv-proyectos-machine-learning

