MAESTRÍA EN CIENCIA DE DATOS

FUNDAMENTOS CIENCIA DE DATOS

PROYECTO #1

TEMA:

“Modelo de Predicción del Consumo de Diésel en Unidades de Transporte Publico**”.**

DESCRIPCIÓN DE LA PROPUESTA:

En el contexto del Transporte Público de Quito, una gestión eficiente del abastecimiento de diésel es crucial para asegurar la operación continua de las unidades. Sin embargo, la planificación del consumo muchas veces se realiza sin considerar patrones históricos o predicciones, lo que puede generar demoras, desperdicio de recursos y descoordinación en el mantenimiento, tanto de unidades de transporte como en surtidores. El problema de investigación es la falta de un modelo que permita anticipar el consumo de diésel por unidad o flota.

Entendimiento de Negocio:

Es posible predecir con un modelo de regresión el consumo diario o semanal de diésel por unidad de transporte público utilizando variables históricas como tipo de unidad, kilometraje y fechas.

Pregunta: ¿Puede un modelo predictivo mejorar la planificación del abastecimiento de diésel en el transporte público de Quito?

Valor de Aporte: El modelo predictivo proporcionará un valor clave al permitir anticipar el consumo de diésel, lo que facilitará la programación del reabastecimiento y mantenimiento, tanto de unidades de transporte como en surtidores, reduciendo costos operativos, optimizando recursos y mejorando la continuidad del servicio. Este enfoque puede escalarse para apoyar la toma de decisiones en la gestión de combustibles a nivel institucional

Entendimiento de los Datos:

Origen: Los datos provienen de los registros históricos del sistema de abastecimiento de combustible de la EPMTPQ, estos datos describen la fecha, el dispensador, hora, el ID de la unidad, tipo de flota, galones y kilometraje.

Variables y su Relevancia: La fecha permite detectar estacionalidad y patrones de consumo de manera diaria, semana, mensual y anual. La unidad y el tipo de flota permite distinguir hábitos de consumo que facilitarán la planificación de mantenimientos preventivos. El kilometraje y los galones nos puede ayudar a determinar anomalías en el consumo de diésel y establecer posibles daños. El dispensador nos ayuda a determinar la frecuencia de abastecimiento.

Posibles Desafíos: Datos faltantes, valores mal registrados en los que incluye inconsistencias en el registro del ID de la unidad, kilometrajes o galones dispensados, duplicidad en los registros, registros incoherentes y formatos incorrectos.

Preparación de los Datos:

Tareas de Limpieza: Eliminación de datos duplicados, rellenar o eliminar datos faltantes y conversión de variables a los formatos correctos para mejorar su manipulación.

Entendimiento del Problema y de los Datos (EDA)

```
raw_diesel = pd.read_csv(r"C:\Users\andre\OneDrive\Escritorio\USFQ\Fundamentos en Ciencia de Datos\Proyeccion_Consumo_Di
raw_diesel.head()
```

✓ 0.4s Python

Se almacena el dataset en la variable raw_diesel para su procesamiento de identificación.

```
raw_diesel.info()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 228209 entries, 0 to 228208
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   FECHA        228209 non-null  object
 1   DISPENSADOR  228209 non-null  int64
 2   HORA         228209 non-null  object
 3   VEHICULO     228207 non-null  object
 4   FLOTA        193574 non-null  object
 5   GALONES      228209 non-null  object
 6   KILOMETRAJE  227988 non-null  object
dtypes: int64(1), object(6)
memory usage: 12.2+ MB
```

Mediante raw_diesel.info() se puede observar que existe un total de 228209 datos ingresados, y que en las columnas pertenecientes a "VEHICULO", "FLOTA", "GALONES" y "KILOMERAJE" presentan valores nulos, también se puede observar que en la "FECHA", "HORA", "GALONES" y "KILOMETRAJE" no tienen los formatos adecuados, ya que estas se almacenan como texto y no como tiempo/fecha y valores flotantes.

```
raw_duplicados = raw_diesel.duplicated().sum()
print('Se encontraron valores duplicados: ', raw_duplicados)
```

✓ 0.2s

```
Se encontraron valores duplicados:  1
```

Se puede observar que se encontró un valor duplicado.

```
print('El dato duplicado es el siguiente: ')
raw_diesel[raw_diesel.duplicated()]
```

✓ 0.1s

```
El dato duplicado es el siguiente:
```

	FECHA	DISPENSADOR	HORA	VEHICULO	FLOTA	GALONES	KILOMETRAJE
23196	3/8/2020	4	10:15:00	T008	TROLEBUS	2,8310	63703,0000

Se detecta el valor duplicado.

	FECHA	DISPENSADOR	HORA	VEHICULO	FLOTA	GALONES	KILOMETRAJE
23195	3/8/2020	4	10:15:00	T008	TROLEBUS	2,8310	63703,0000
23196	3/8/2020	4	10:15:00	T008	TROLEBUS	2,8310	63703,0000

```
raw_diesel.isna().sum()
```
✓ 0.0s

```
FECHA              0
DISPENSADOR        0
HORA               0
VEHICULO           2
FLOTA          34635
GALONES            0
KILOMETRAJE      221
dtype: int64
```

Se procede a verificar la cantidad de valores faltantes.

```
ausentes_vh = raw_diesel[raw_diesel.isna()['VEHICULO']]
ausentes_vh
```
✓ 0.0s

	FECHA	DISPENSADOR	HORA	VEHICULO	FLOTA	GALONES	KILOMETRAJE
147742	1/8/2022	6	0:00:00	NaN	NaN	0,0000	0,0000
172389	11/12/2022	4	18:29:00	NaN	NaN	0,0010	0,0000

Como se puede observar en la columna "VEHICULO" se encuentran datos vacíos, y tomando en cuenta la fila de "GALONES" y "KILOMETRAJE" pueden ser errores en los registros, los cuales al ser eliminados no afectan al estudio ya que los valores que registran tanto en galones como kilometraje son mínimos.

```
ausentes_flt = raw_diesel[raw_diesel.isna()['FLOTA']]
ausentes_flt['VEHICULO'].unique()
```
✓ 0.0s

```
array(['M036', 'M026', 'D015', 'D003', 'D025', 'T111', 'M020', 'D009',
       'D030', 'D042', 'D002', 'T095', 'T033', 'M019', 'S056', 'B036',
       'B038', 'T051', 'A114', 'B045', 'T028', 'T042', 'T005', 'T043',
       'T003', 'D029', 'D006', 'T038', 'T029', 'D038', 'M040', 'T032',
       'D020', 't029', 't111', 'T81', 'D004', 'D016', 'T045', 'D024',
       'T048', 'D021', 'T019', 'T053', 'D034', 'T075', 'D033', 'T031',
       'D010', 'A117', 'T088', 'T015', 'T037', 'COMPRESOR', 'T034',
```

Se puede observar que existen varios valores que no describen un tipo de flota, en su mayoría son errores al momento de registrar las placas de los vehículos, como la "t029" que en el registro real si existe una "T029", también se puede observar que existen el registro de unidades que empiezan con una nomenclatura "PMA" o "P", esto se presenta debido a que no son unidades pertenecientes a la flota Trolebus. También se puede observar que existen unidades que han sido dadas de baja en tiempo recientes, por lo que no se encuentran registrados en una flota específica.

```
ausentes_km = raw_diesel[raw_diesel.isna()['KILOMETRAJE']]
ausentes_km['VEHICULO'].unique()
```

✓ 0.0s

```
array(['LAVTALLER', 'T073', 'D006', 'M038', 'COMPRESOR', 'A114', 'B032',
       'M014', 'T106', 'T017', 'T004', 'PMA3053', 'PMA3026', 'PMA7672',
       'PMA3322', 'PMA3249', 'PMA3066', 'PMF780', 'PMA3325', 'PMA8561',
       'PMA706', 'P-600789561', 'PMA7388', 'PMA3050', 'PMA7394',
       'PMA3039', 'PMA7717', 'PMA3267', 'D035', 'PMA7721', 'D037',
       'P-G00789561', 'PMA3237', 'PMA7230', 'PMA8558', 'lavtaller',
       'PMA7684', 'PMF0782', 'M011', 'B052', 'PMA7164', 'PMA3093',
       'PMF0872', 'PMA7561', 'PME0782', 'PMA8567', 'PMA7113', 'PMA7689',
       'PMA7878', 'PMA3239', 'PMA7108', 'PMA7699', 'PMD0501', 'P-30162',
       'PMA7032', 'PMA7713', 'PMA8564', 'PMA7109', 'S008', 'PMA7393',
       'V017', 'V060', 'T027'], dtype=object)
```

Los datos de "COMPRESOR" y "LAVTALLER" son implementos de trabajo en los que se utiliza Diesel y no tienen recorrido en kilómetros, las unidades que empiezan con "P" o "PMA" son unidades de trabajo no pertenecientes a la empresa por lo que no se registra el kilometraje, las unidades "T", "D", "M", "B" y "A", se completarán los datos con la media del recorrido de las unidades y el último registro.

Data Wrangling

```
raw_diesel = raw_diesel.drop_duplicates()
raw_diesel.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 228208 entries, 0 to 228208
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype 
---  ------       --------------   ----- 
 0   FECHA        228208 non-null  object
 1   DISPENSADOR  228208 non-null  int64 
 2   HORA         228201 non-null  object
 3   VEHICULO     228206 non-null  object
 4   FLOTA        193573 non-null  object
 5   GALONES      228208 non-null  object
 6   KILOMETRAJE  227987 non-null  object
dtypes: int64(1), object(6)
memory usage: 13.9+ MB
```

Descartamos valores duplicados.

```
raw_diesel = raw_diesel.dropna(subset=['FECHA', 'DISPENSADOR', 'HORA', 'VEHICULO', 'GALONES', 'KILOMETRAJE', 'FLOTA'])
raw_diesel.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 193437 entries, 0 to 228208
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype 
---  ------       --------------   ----- 
 0   FECHA        193437 non-null  object
 1   DISPENSADOR  193437 non-null  int64 
 2   HORA         193437 non-null  object
 3   VEHICULO     193437 non-null  object
 4   FLOTA        193437 non-null  object
 5   GALONES      193437 non-null  object
 6   KILOMETRAJE  193437 non-null  object
dtypes: int64(1), object(6)
memory usage: 11.8+ MB
```

Eliminamos valores nulos en la cada una de las columnas del dataset.

```
raw_diesel['KILOMETRAJE'] = raw_diesel['KILOMETRAJE'].astype(str).str.replace('.', '', regex=False).str.replace(',', '.'
raw_diesel['KILOMETRAJE'] = pd.to_numeric(raw_diesel['KILOMETRAJE'], errors='coerce')
```

Se realizaron varios reemplazos en los valores de Kilometraje ya que estos contenían varios formatos distintos, incluían separadores de miles (9.9587,12) y los números de tipo flotante utilizaban comas en vez de puntos (198,56); lo que dificultaba la limpieza de los datos.

```
raw_diesel.info()
✓ 0.0s
<class 'pandas.core.frame.DataFrame'>
Index: 193437 entries, 0 to 228208
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   FECHA        193437 non-null  object
 1   DISPENSADOR  193437 non-null  int64
 2   HORA         193437 non-null  object
 3   VEHICULO     193437 non-null  object
 4   FLOTA        193437 non-null  object
 5   GALONES      193437 non-null  object
 6   KILOMETRAJE  193391 non-null  float64
dtypes: float64(1), int64(1), object(5)
memory usage: 11.8+ MB
```

Podemos apreciar que, de los 228208 registros, se lograron mantener 193391 registros.

```
raw_diesel = raw_diesel.dropna(subset=['KILOMETRAJE', 'DISPENSADOR', 'GALONES'])
raw_diesel.info()
✓ 0.0s
<class 'pandas.core.frame.DataFrame'>
Index: 193391 entries, 0 to 228208
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   FECHA        193391 non-null  object
 1   DISPENSADOR  193391 non-null  int64
 2   HORA         193391 non-null  object
 3   VEHICULO     193391 non-null  object
 4   FLOTA        193391 non-null  object
 5   GALONES      193391 non-null  object
 6   KILOMETRAJE  193391 non-null  float64
dtypes: float64(1), int64(1), object(5)
memory usage: 11.8+ MB
```

Se comprueba nuevamente de que no existan valores duplicados.

```
raw_diesel['FECHA'] = pd.to_datetime(raw_diesel['FECHA'], errors='coerce')
raw_diesel = raw_diesel.dropna(subset=['FECHA'])
raw_diesel.info()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
Index: 193391 entries, 0 to 228208
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   FECHA        193391 non-null  datetime64[ns]
 1   DISPENSADOR  193391 non-null  int64
 2   HORA         193391 non-null  object
 3   VEHICULO     193391 non-null  object
 4   FLOTA        193391 non-null  object
 5   GALONES      193391 non-null  object
 6   KILOMETRAJE  193391 non-null  float64
dtypes: datetime64[ns](1), float64(1), int64(1), object(4)
memory usage: 11.8+ MB
```

Se comprueba nuevamente de que no existan valores duplicados.

```
raw_diesel['AÑO'] = raw_diesel['FECHA'].dt.year
raw_diesel
```

✓ 0.0s

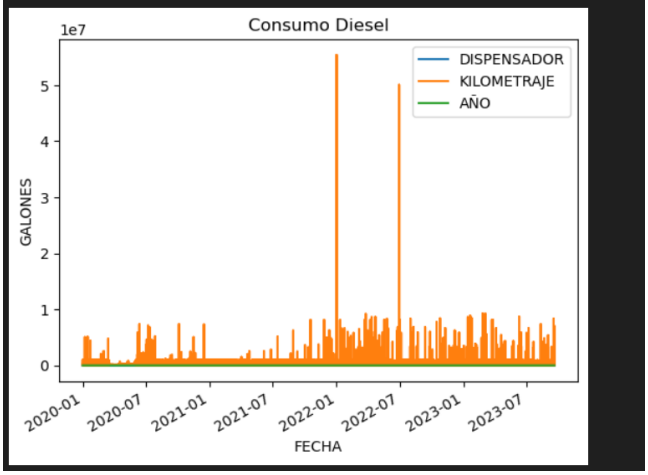
	FECHA	DISPENSADOR	HORA	VEHICULO	FLOTA	GALONES	KILOMETRAJE	AÑO
0	2020-01-01	1	10:35:00	S024	B12M	53,568	411475.8	2020
1	2020-01-01	1	12:12:00	S029	B12M	54,197	479353.4	2020
2	2020-01-01	1	12:20:00	S011	B12M	56,556	463696.8	2020
3	2020-01-01	1	13:10:00	S074	B12M	65,614	451279.5	2020
4	2020-01-01	1	13:30:00	S015	B12M	42,879	418781.5	2020
...	...	...	...	...	...	...	...	...
228204	2023-09-20	6	13:36:00	V064	BI ARTICULADOS	47,218	202419.3	2023
228205	2023-09-20	6	14:20:00	V017	BI ARTICULADOS	86,447	371091.3	2023
228206	2023-09-20	6	15:07:00	S080	B12M	66,144	642887.3	2023

Se crea una nueva columna de datos nombrada “AÑO”, la cual nos permite identificar valores en el transcurso del tiempo.

```
raw_diesel.plot(title= 'Consumo Diesel', xlabel= 'FECHA', ylabel='GALONES')
```

✓ 1.0s

```
<Axes: title={'center': 'Consumo Diesel'}, xlabel='FECHA', ylabel='GALONES'>
```

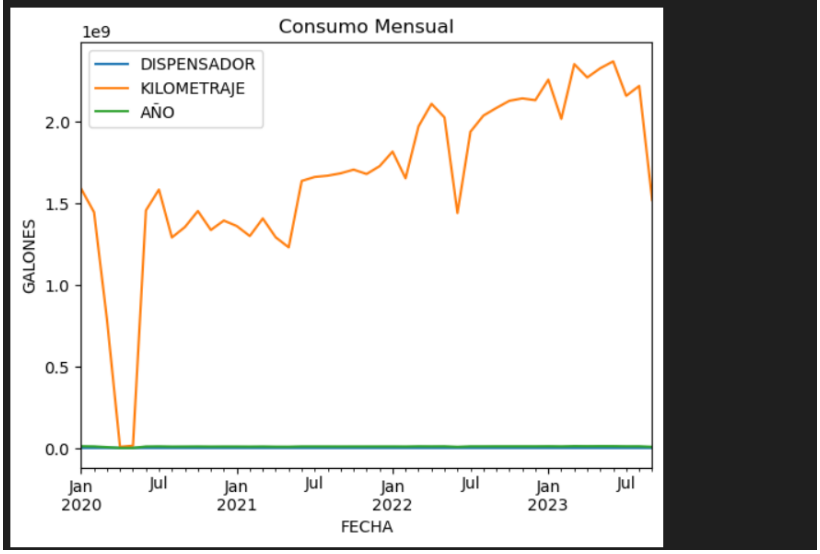


Se puede apreciar un consumo excesivo de galones, los cuales demuestra un patrón anormal.

```
raw_diesel_mensual = raw_diesel.resample('1ME').sum()
raw_diesel_mensual.plot(title = 'Consumo Mensual', xlabel = 'FECHA', ylabel = 'GALONES')
```

✓ 0.7s

```
<Axes: title={'center': 'Consumo Mensual'}, xlabel='FECHA', ylabel='GALONES'>
```



Se reconstruye la muestra y podemos identificar un patrón de consumo el cual se encuentra expresado en meses.

Estructura del Repositorio de GitHub

```
Proyeccion_Consumo_Diesel/
│
├── data/                                  # Datos en crudo y limpios
│   ├── Consumo_Diesel.csv                 # Dataset original
│   └── Consumo_Diesel_limpio.csv          # Dataset limpio
│
├── notebooks/                             # Notebooks de análisis
│   └── EDA_Wrangling_Diesel.ipynb
│
├── reports/                               # Reportes o presentaciones
│   └── Reporte_1.pdf
│
├── .gitignore              # Ignorar archivos innecesarios (ej. *.pyc, .ipynb_checkpoints)
└── README.md               # Descripción del proyecto
```