Laboratorio 2 – Proyecto de Predicción de Predicción de Bicicletas

En esta sección ud. estará a cargo de evaluar como se puede realizar la predicción de bicicletas de un local de alquiler. Como científico de datos tiene la labor de predecir la siguiente demanda del local por medio de un modelo de machine learning utilizando scikit-learn.

Copiando el Dataset al Computador

Primero debemos adquirir el dataset. Nuestro dataset a descargar está en la siguiente dirección:

- https://archive.ics.uci.edu/ml/machine-learning-databases/00275/Bike-Sharing-Dataset.zip

Descargar los datos

Haremos una función para descargar los datos de la web, esto es muy útil para futuro.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os
from six.moves import urllib

DS_EXTRACT_PATH = "/content/drive/MyDrive/Laboratorio No2"
DS_URL = "https://archive.ics.uci.edu/ml/machine-learning-databases/00275/Bike-Sharing-Dataset.zip"

def download_dataset(dataset_url=DS_URL, dataset_pc_path=DS_EXTRACT_PATH):
    if not os.path.isdir(dataset_pc_path):
        os.makedirs(dataset_pc_path)
    parts = os.path.split(dataset_url)
    zipname = parts[-1]
    zip_path = os.path.join(dataset_pc_path, zipname)
    urllib.request.urlretrieve(dataset_url, zip_path)


# Salida Esperada:
# Dataset en formato zip descargado en: <esta_carpeta>/datasets/capitalbikeshare/<nombre_del_zip>
download_dataset(DS_URL)
```

Saving...

Haremos una función para descomprimir el dataset en esta carpeta y finalmente borraremos el zip file.

```
from zipfile import ZipFile
import numpy as np

def unzip_dataset(dataset_pc_path=DS_EXTRACT_PATH):
    zip_path = os.path.join(dataset_pc_path, "Bike-Sharing-Dataset.zip")
    with ZipFile(zip_path, 'r') as zip_ref:
```

```
        zip_ref.extractall(dataset_pc_path)


unzip_dataset(DS_EXTRACT_PATH)
## Salida Esperada:
# Dataset en formato csv descomprimodo en: <esta_carpeta>/datasets/capitalbikeshare/
# archivos: hour.csv, day.csv, Readme.txt
```

Cargar y explorar el dataset

Ahora cargue el dataset y explore las diferentes variables del dataset:

- hour.csv

Cargar el dataset con pandas

```
import pandas as pd

def load_bike_hourly_data(bike_dataset_path=DS_EXTRACT_PATH):
    data_path = os.path.join(bike_dataset_path, "hour.csv")
    df = pd.read_csv(data_path)
    return df
```

Explorar las primeras 48 filas

```
bikeshare = load_bike_hourly_data()
bikeshare.head(48)
```

29	30	2011-01-02	1	0	1	6	0	0	0
30	31	2011-01-02	1	0	1	7	0	0	0
31	32	2011-01-02	1	0	1	8	0	0	0
32	33	2011-01-02	1	0	1	9	0	0	0
33	34	2011-01-02	1	0	1	10	0	0	0
34	35	2011-01-02	1	0	1	11	0	0	0
35	36	2011-01-02	1	0	1	12	0	0	0
36	37	2011-01-02	1	0	1	13	0	0	0
37	38	2011-01-02	1	0	1	14	0	0	0
38	39	2011-01-02	1	0	1	15	0	0	0
39	40	2011-01-02	1	0	1	16	0	0	0
40	41	2011-01-02	1	0	1	17	0	0	0
41	42	2011-01-02	1	0	1	18	0	0	0
42	43	2011-01-02	1	0	1	19	0	0	0
43	44	2011-01-02	1	0	1	20	0	0	0
44	45	2011-01-02	1	0	1	21	0	0	0
45	46	2011-01-02	1	0	1	22	0	0	0
46	47	2011-01-02	1	0	1	23	0	0	0
					)	1	0	1	1

Saving...

Desplegar información del dataset

dataset

Saving...

```
bikeshare.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17379 entries, 0 to 17378
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   instant     17379 non-null  int64
 1   dteday      17379 non-null  object
 2   season      17379 non-null  int64
 3   yr          17379 non-null  int64
```

```
 4   mnth        17379 non-null  int64
 5   hr          17379 non-null  int64
 6   holiday     17379 non-null  int64
 7   weekday     17379 non-null  int64
 8   workingday  17379 non-null  int64
 9   weathersit  17379 non-null  int64
 10  temp        17379 non-null  float64
 11  atemp       17379 non-null  float64
 12  hum         17379 non-null  float64
 13  windspeed   17379 non-null  float64
 14  casual      17379 non-null  int64
 15  registered  17379 non-null  int64
 16  cnt         17379 non-null  int64
dtypes: float64(4), int64(12), object(1)
memory usage: 2.3+ MB
```

Visualizando el dataset

El dataset, como observó, tiene el número de personas para cada hora del día desde Enero del 2011 hasta Diciembre 2017. El número de personas que manejan se divide entre casuales y registrados sumados en la columna 'cnt'.

- Grafique el número de manejadores de bicicleta de los primeros 10 días del dataset. Observe el dataset y note que el día se divide en 24 horas. Puede ver la cantidad de rentas por horas allí.

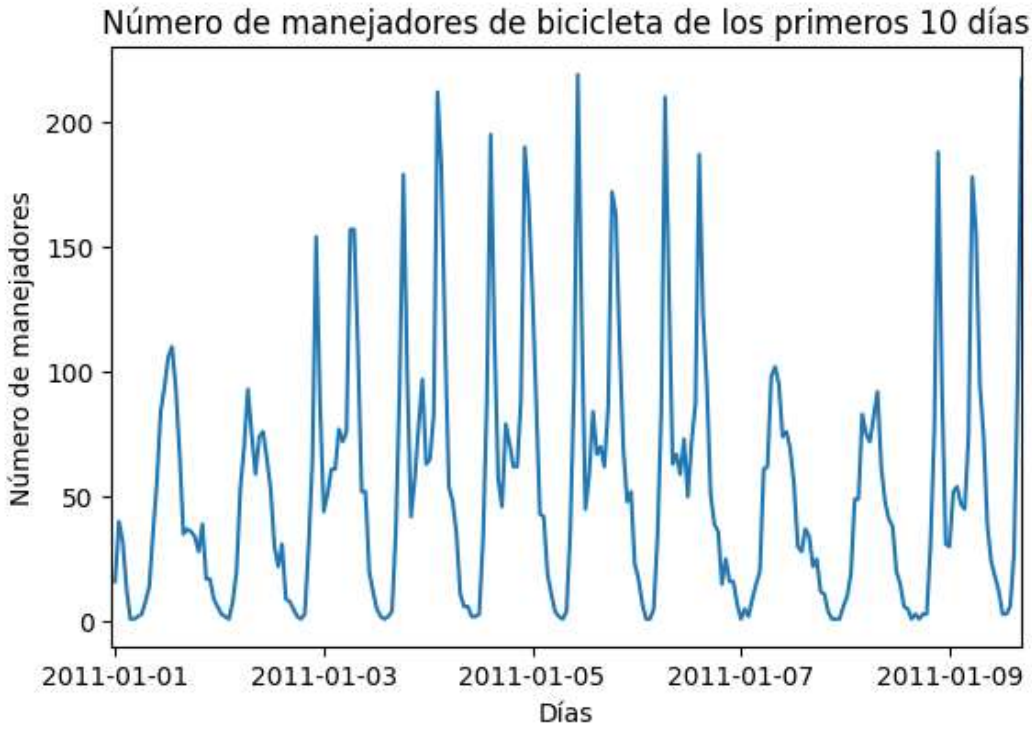
El dataset es difícil de analizar normalmente, tiene altibajos, en especial cuando las personas van del trabajo a la casa y los fines de semana. De las columnas anteriores también tenemos velocidad de viento, humedad, temperatura. Todo esto afecta el alquiler de las bicicletas. Trataremos de capturar estas características con un modelo.

```
import matplotlib.pyplot as plt

# Obtener los primeros 10 días del dataset
primeros_10_dias = bikeshare.head(240)
horas = range(1, 241)
manejadores = primeros_10_dias['cnt']

fechas_completas = primeros_10_dias['dteday'].unique()[:10]
fechas_impares = fechas_completas[::2]

ticks = horas[::55][:len(fechas_impares)]
labels = fechas_impares.tolist()

plt.plot(horas, manejadores)
plt.xlabel('Días')
```

Saving...

```
bicicleta de los primeros 10 días')

plt.xlim(0, max(horas))  # Establecer límites del eje x desde 0 hasta el valor máximo de las horas

plt.xticks(ticks)
plt.gca().set_xticklabels(labels, rotation=0, ha='center')

# Ajustar manualmente la posición de las etiquetas
ax = plt.gca()


plt.subplots_adjust(bottom=0.2)
```

```
plt.show()
```

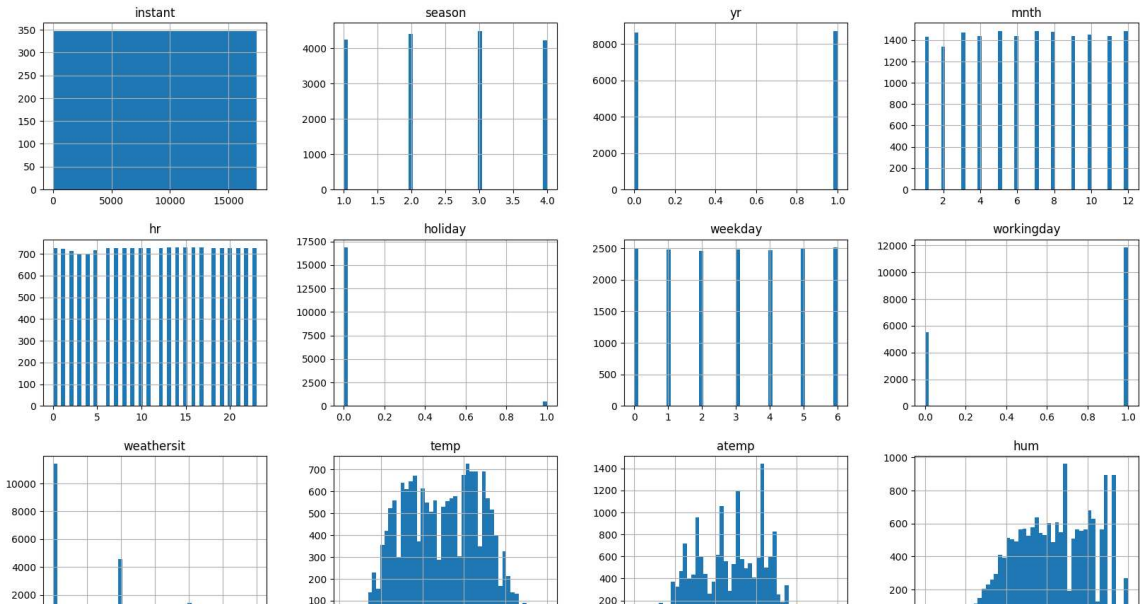


Gane más información de la distribución del dataset

Simplemente grafique con hist() para verificar/validar la distribución de los datos y ganar más información del dataset

```
# Graficamos la informacion de los datos en Histogramas

# Importamos la libreria Matplotlib para realizar graficias
from matplotlib import pyplot

# Llamamos la variable de datos, indicamos el tipo de grafica.
# El argumento bins nos permite establecer un rango de datos para distribuirlos en graficas de hist
# Figuresize nos define las dimensiones de estos graficos
bikeshare.hist(bins=50, figsize=(20,15))

# Mostramos los datos
plt.show()
```

Saving...



Variables Dummy - Transformación de Data Categórica

Los datos categóricos como temporada, mes, y año deben ser transformados a números debido a que los modelos solo trabajan con esto. También tendremos variables binarias. La acción la realizaremos com pandas y la función:

- get_dummies()
- columnas dummy = 'season', 'weathersit', 'mnth', 'hr', 'weekday'
- columnas a borrar = 'instant', 'dteday', 'season', 'weathersit', 'weekday', 'atemp', 'mnth', 'workingday', 'hr'

```
# Columnas para realizar la codificación one-hot
dummy_cols = ['season', 'weathersit', 'weekday', 'mnth', 'hr']

# Realizar codificación one-hot para cada columna
for each in dummy_cols:
    dummies = pd.get_dummies(bikeshare[each], prefix=each, drop_first=False)
    bikeshare = pd.concat([bikeshare, dummies], axis=1)

# Columnas a eliminar
drop_cols = ['instant', 'dteday', 'season', 'weathersit', 'weekday', 'atemp', 'mnth', 'workingday',

# Eliminar columnas
data = bikeshare.drop(drop_cols, axis=1)

# Mostrar las primeras filas del dataframe resultante
```

Saving...

	yr	holiday	temp	hum	windspeed	casual	registered	cnt	season_1	season_2
0	0	0	0.24	0.81	0.0	3	13	16	1	0
1	0	0	0.22	0.80	0.0	8	32	40	1	0
4	0	0	0.24	0.75	0.0	0	1	1	1	0

Escalar el Dataset

- variables a escalar = 'casual', 'registered', 'cnt', 'temp', 'hum', 'windspeed'

```
to_scale = ['casual', 'registered', 'cnt', 'temp', 'hum', 'windspeed']
# Store scalings in a dictionary so we can convert back later
scaled_features = {}
for each in to_scale:
    mean, std = data[each].mean(), data[each].std()
    scaled_features[each] = [mean, std]
    data.loc[:, each] = (data[each] - mean)/std
```

Separación del Dataset

Separar el dataset en Training y Test

- 20% test set
- Campos de Test = 'cnt', 'casual', 'registered'
- Campos de Train = todos los demas

```
# Separación del Dataset
from sklearn.model_selection import train_test_split

test_fields = ['cnt', 'casual', 'registered']
train_fields = [col for col in data.columns if col not in test_fields]

X_train, X_test, y_train, y_test = train_test_split(data[train_fields], data[test_fields], test_siz
```

Entrenamiento

Entrenar el modelo basado en el regresor KNeighobrsRegressor

- n_neighbors = 4
- n_jobs = 4

Saving...

```
borsRegressor
from sklearn.metrics import mean_squared_error

n_neighbors = 4
n_jobs = 4

knn = KNeighborsRegressor(n_neighbors=n_neighbors, n_jobs=n_jobs)
knn.fit(X_train, y_train)
```

Guardar el Modelo

Una parte importante es que luego de entrenar podríamos guardar el modelo. Este paso se realiza luego de haber evaluado varios y validar que cumple con las métricas. A manera de prueba lo realizaremos antes, sin embargo recordar que es parte de los últimos pasos.

```
# Guardar el Modelo
import pickle

model_filename = 'knn_model.pkl'
pickle.dump(knn, open(model_filename, 'wb'))
```

Cargar el Modelo

Cuando poseemos la aplicación final, por ejemplo una aplicación por celular o una aplicación web, podemos 'consumir' el modelo llamandolo para realizar nuestras predicciones. Para esto el modelo debe estar disponible desde una ubicación conocida en la máquina o dispositivo a ejecutar.

```
# Cargar el Modelo
loaded_model = pickle.load(open(model_filename, 'rb'))
```

Realizar predicciones

Realizar las predicciones con el modelo de los datos de test

```
# Realizar predicciones
predictions = loaded_model.predict(X_test)
```

Evaluar el modelo por medio de métricas

Evaluar el modelo por medio de las métricas de error medio cuadrado

```
# Evaluar el modelo por medio de métricas
mse = mean_squared_error(y_test, predictions)
mse
```


Graficar los resultados

Graficar los resultados de la predicción vs los resultados de las pruebas

```
# Graficamos los resultados
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

fig, ax = plt.subplots(figsize=(20, 15))
```