Modelo ML

1st Daniel Alejandro Peñaloza Cuesta
Universidad Distrital Franciso Jose de Caldas
Facultad de ingeniería
Bogotá, Colombia
dapenalozac@udistrital.edu.co

***Abstract*—This project presents a regression-oriented machine learning model, implemented using Random Forest Regressor technique from scikit-learn library in Python. The purpose of the model is to estimate and fill in missing values in the city's air pollutant reports. Through a ser of multiple decision trees, the model learns patterns from the available data and predicts the missing values with greater robustness and accuracy than traditional methods. This approach helps improve the continuity, consistency, and reliability of the environmental databases used for further analysis.**

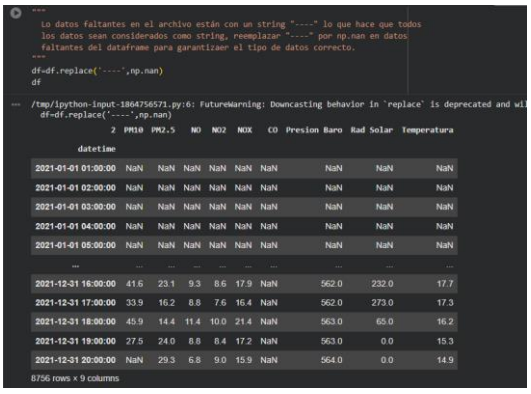
***Index Terms*—machine learning, library, Phyton, Pollutant**

I. Introduction

Este proyecto es la continuación de una investigación que se hizo de los contaminantes atmosféricos, con este proyecto se busca hacer una predicción espacio-temporal para tener un pronostico de toda la ciudad. Para esto es necesario de los reportes de las 19 estaciones de monitoreo, pero se evidencio que estos reportes tienen vacíos muy grandes de datos (incluso meses), al ser tan notorio esto no es viable una interpolación, sino que hace falta completar los datos mediante un modelo de regresión lineal de ML (machine learning), para ahí si hacer un modelo cnn-lstm para una predicción a escala.

II. PROCEDIMIENTO

para iniciar se descargó los reportes de todas las estaciones para mediante un código en Google collab se editaron estos reportes para poderlos usar en el modelo de regresión lineal. Los cambios que se le hicieron a los reportes fueron cambiar sus índices, pasar de rectangular a polar unos datos, acomodar los datos vacíos para que quedaran en formato Nan y se pudieran hacer operaciones con los dataframes, también se seleccionaron los datos que se encontraban en los reportes de todas las estaciones. Por último, fue guardar todo esto en archivos .csv para poder hacer el modelo.



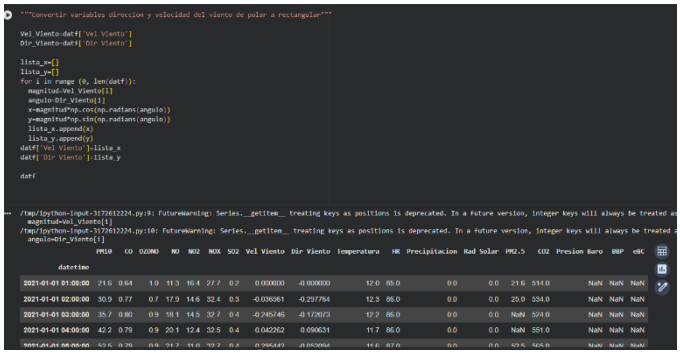
```
"""
 Lo datos faltantes en el archivo están con un string "----" lo que hace que todos
 los datos sean considerados como string, reemplazar "----" por np.nan en datos
 faltantes del dataframe para garantizaer el tipo de datos correcto.
"""
df=df.replace('----',np.nan)
df
```

```
/tmp/ipython-input-1864756571.py:6: FutureWarning: Downcasting behavior in `replace` is deprecated and wi
  df=df.replace('----',np.nan)
```

datetime	2	PM10	PM2.5	NO	NO2	NOX	CO	Presion Baro	Rad Solar	Temperatura
2021-01-01 01:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2021-01-01 02:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2021-01-01 03:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2021-01-01 04:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2021-01-01 05:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
2021-12-31 16:00:00	41.6	23.1	9.3	8.6	17.9	NaN	562.0	232.0	17.7	
2021-12-31 17:00:00	33.9	16.2	8.8	7.6	16.4	NaN	562.0	273.0	17.3	
2021-12-31 18:00:00	45.9	14.4	11.4	10.0	21.4	NaN	563.0	65.0	16.2	
2021-12-31 19:00:00	27.5	24.0	8.8	8.4	17.2	NaN	563.0	0.0	15.3	
2021-12-31 20:00:00	NaN	29.3	6.8	9.0	15.9	NaN	564.0	0.0	14.9	

8756 rows × 9 columns

Se hace el cambio de string a Nan.



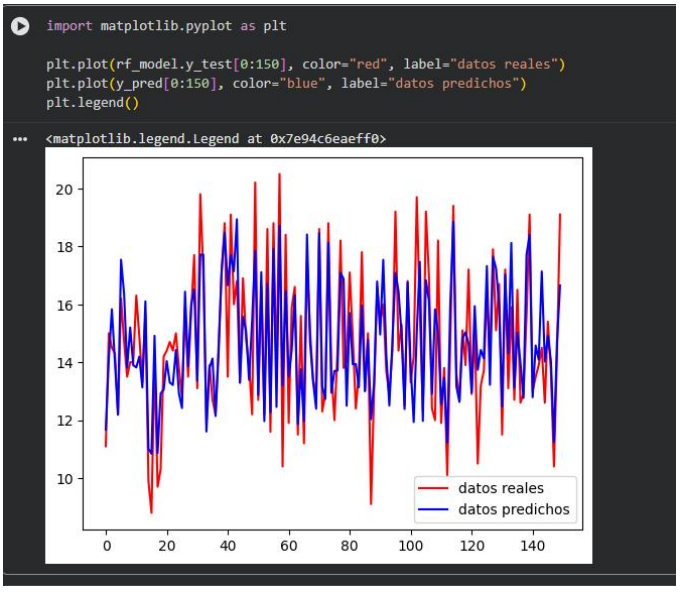
```
"""Convertir variables dirección y velocidad del viento de polar a rectangular"""

Vel_Viento=datf['Vel Viento']
Dir_Viento=datf['Dir Viento']

lista_x=[]
lista_y=[]
for i in range (0, len(datf)):
  magnitud=Vel_Viento[i]
  angulo=Dir_Viento[i]
  x=magnitud*np.cos(np.radians(angulo))
  y=magnitud*np.sin(np.radians(angulo))
  lista_x.append(x)
  lista_y.append(y)
datf['Vel Viento']=lista_x
datf['Dir Viento']=lista_y

datf
```

```
/tmp/ipython-input-3172612224.py:9: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as
  magnitud=Vel_Viento[i]
/tmp/ipython-input-3172612224.py:10: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated
  angulo=Dir_Viento[i]
```

datetime	PM10	CO	O3DNO	NO	NO2	NOX	SO2	Vel Viento	Dir Viento	Temperatura	HR	Precipitacion	Rad Solar	PM2.5	CO2	Presion Baro	BBP	eBC
2021-01-01 01:00:00	21.6	0.64	1.0	11.3	16.4	27.7	0.2	0.000000	-0.000000	12.0	85.0	0.0	0.0	21.6	534.0	NaN	NaN	NaN
2021-01-01 02:00:00	30.9	0.77	0.7	17.9	14.6	32.4	0.3	-0.036961	-0.297764	12.3	86.0	0.0	0.0	25.0	534.0	NaN	NaN	NaN
2021-01-01 03:00:00	35.7	0.80	0.9	18.1	14.5	32.7	0.4	-0.245746	-0.172073	12.2	86.0	0.0	0.0	NaN	524.0	NaN	NaN	NaN
2021-01-01 04:00:00	42.2	0.79	0.9	20.1	12.4	32.5	0.4	0.042262	0.090631	11.7	86.0	0.0	0.0	NaN	551.0	NaN	NaN	NaN

Cambio de polar a rectangular.

Después de este se hace el Código del modelo ML en este se toman los archivos csv y se escogen los datos que estén en todos los reportes o en la mayoría, se monta el modelo mediante bosques aleatorios con la librería scikit-learn en Python. después de esto se entrena el modelo para después hacer las predicciones y ver mediante porcentajes y graficas que tan eficiente fue la predicción de dichos datos y guardar esos datos predichos en un csv.



```
import matplotlib.pyplot as plt

plt.plot(rf_model.y_test[0:150], color="red", label="datos reales")
plt.plot(y_pred[0:150], color="blue", label="datos predichos")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7e94c6eaeff0>
```


Esta es la gráfica de comparación, para validar visualmente que tan eficiente fue la predicción.

```
rf_model = RandomForestModel(n_estimators=350, max_depth=50, random_state=12, test_size=0.2)

# Dividir los datos
rf_model.split_data(X, y)

# Entrenar el modelo
rf_model.train()

# Hacer predicciones
y_pred = rf_model.predict()

# Evaluar
r2 = rf_model.evaluate()

Modelo entrenado correctamente.
R² score: 0.690657158780934
```

Esta es la imagen donde se hace uso del modelo y se muestra el porcentaje de eficiencia.

Finalmente se recopila los datos predichos en un archivo csv para completar los reportes de estaciones y así poder continuar con el modelo de la predicción a escala.

CONCLUSIONES

1. La limpieza, estandarización y transformación de los datos fueron etapas fundamentales para el desempeño del modelo. El cambio de coordenadas polares a rectangulares, la unificación de índices y la correcta identificación de valores faltantes (Nan) permitieron crear una base de datos apta para el entrenamiento y mejoraron la calidad de las predicciones
2. La función Random Forest Regressor demostró ser una herramienta eficaz ya que tiene variedad de parámetros para poder hacer más robusto el modelo.
3. Gracias a la automatización del proceso en Google colab y el uso de Python, se logró una metodología reproducible, escalable y adaptable para futuras fases del proyecto. Esto facilitara la integración con modelos más avanzados de predicción atmosférica