SISTEMA DE PRONÓSTICO DE PM10 DE SANTA ANITA – LIMA PARA EL PERIODO 2023 APLICANDO INTELIGENCIA ARTIFICIAL

Olimpio Solis[1], Daniela Jáuregui[2], Alejandra Solis[2]

[1] Organismo de Evaluación y Fiscalización Ambiental (OEFA), Lima, Perú, osolis@oefa.gob.pe
[2] Continental University of Florida, Florida, EEUU, mariadanielajau@gmail.com
[3] pontificia Universidad Católica del Perú (PUCP), Lima, Perú, a20210710@pucp.edu.pe

I. Resumen

El crecimiento urbano descontrolado, la contaminación atmosférica y la escasez de espacios públicos abiertos persisten en las ciudades, la contaminación deteriora la salud de la población y afecta a la productividad de los trabajadores y por tanto, a la economía, siendo esto uno de los puntos principales que promuevan los Objetivos de Desarrollo Sostenible (ODS), en el área del Objetivo 11: Ciudades y Comunidades Sostenibles (Ref. Virtual: https://www.un.org/sustainabledevelopment/es/cities/).

La Secretaría de Gobierno y Transformación Digital de la PCM, en alianza con la Organización de los Estados Americanos (OEA) y la Iniciativa Latinoamericana por los Datos Abiertos (ILDA), Organizan el evento datatón Exprésate Perú con Datos 2024, que es un espacio de innovación que busca empoderar a la ciudadanía en el uso datos gubernamentales para proponer soluciones y hallazgos a través del uso de la inteligencia artificial y el procesamiento de datos. (Ref. Virtual: https://www.gob.pe/expresateperu2024)

Bajo este contexto y tomando como base el Objetivo 11: Ciudades y Comunidades Sostenibles del Desarrollo Sostenible (ODS), hemos desarrollado un modelo automático aplicando inteligencia artificial sobre el pronóstico de material particulado PM10 registrado por la estación de Calidad del Aire de Santa Anita (SENAMHI).

II. Objetivos del Proyecto

- Desarrollar un modelo automático de pronostico de pm10 de una estación de calidad del aire Santa Anita con la finalidad de prevenir a las empresas generadoras de la contaminación que tomen medidas correctivas para disminuir la contaminación de la población del Distrito de Santa Anita
- Disponer de una herramienta a las entidades del Estado que evalúan, supervisan y fiscalizan la contaminación, de esta manera les sirva como una herramienta de Gestión de la vigilancia de la Contaminación del Aire, que bien puede ser utilizada como plan piloto para ser reproducida en otras ciudades.

III. Metodología

El proyecto realiza los siguientes pasos: Captura y Limpieza de datos, Completación de Datos, desarrollo del Modelo de Pronostico aplicando Inteligencia Artificial, Validación del Modelo, Publicación de Resultados.

IV. Conclusiones y Resultados

- Se obtuvo un buen Coeficiente de determinación de 0.91, entre los datos observados y los pronosticados.
- El pronostico fue para las siguientes 24 horas, dando buenos resultados, tomando solo la serie univariada del PM10 para el periodo del 2023.
- Se recomienda mejorar el modelo, incorporando la componente de datos meteorológicos como Temperatura, Humedad y Viento.

Figura 1: Grafico del Modelo de Pronostico de PM10 de anata Anita para 2023

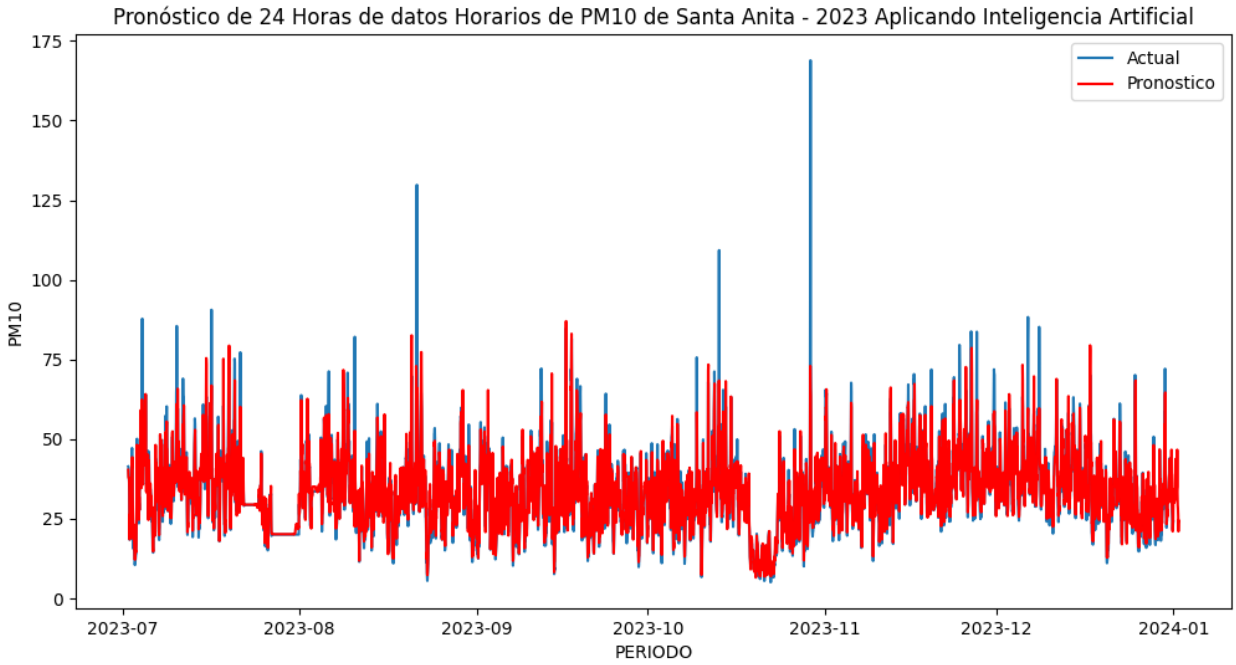
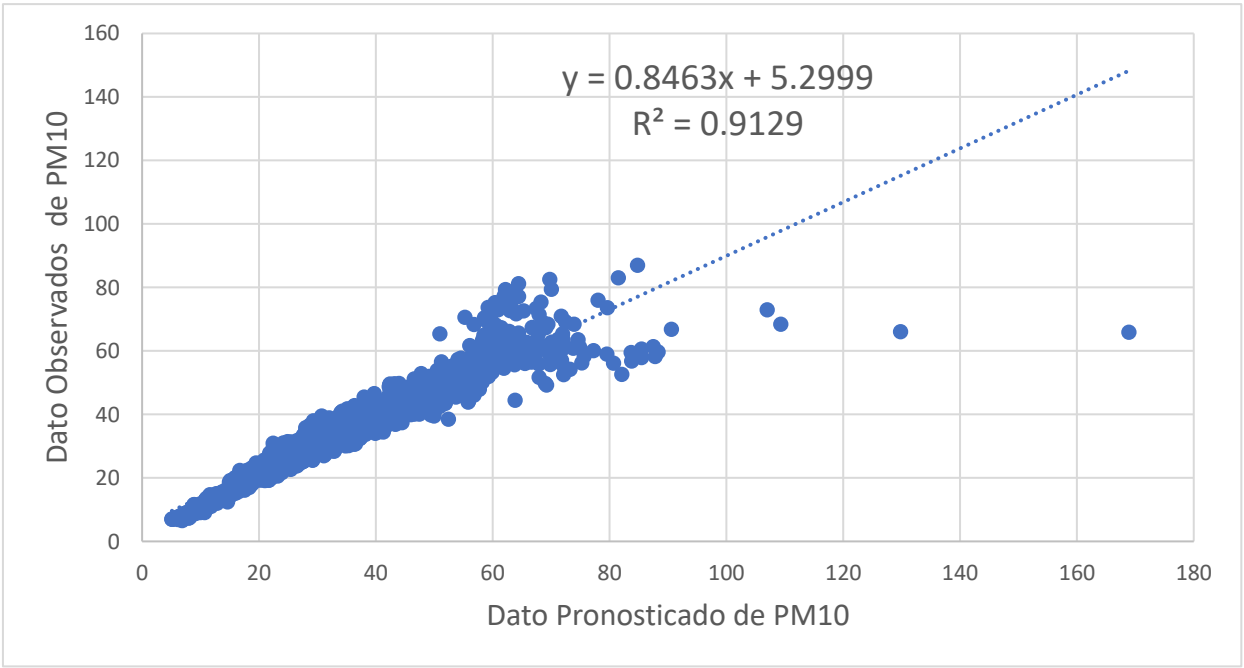


Figura 2: Grafico de Análisis de Correlación de datos Observados y Pronosticados de PM10



Fuente de datos: https://expresateperu.datosabiertos.gob.pe/dataset/monitoreo-de-los-contaminantes-del-aire-en-lima-metropolitana
🚀Datos Abiertos: https://bit.ly/3EpfQhm
🥽Link del Sistema: https://github.com/olimpiosoliscaceres/flask_pronostico

V. Documentación del modelo de pronostico

Este programa realiza un modelo de pronóstico de la concentración de PM10 (partículas finas en el aire) utilizando un modelo de Random Forest en Python. A continuación, se explica cada sección del código y su función en el proceso:

1. Leer el archivo de datos y obtener el rango de fechas completo

- **Lectura de datos**: Se carga un archivo CSV (`dato_ca.csv`) que contiene una serie temporal de datos de PM10. La columna de fechas se llama `PERIODO` y se convierte en el índice del DataFrame para un manejo más eficiente de la serie temporal.

```
df = pd.read_csv(lv_nom_file_origen, sep=';', parse_dates=['PERIODO'])
df['PERIODO'] = pd.to_datetime(df['PERIODO'], format='%Y-%m-%d %H')
df.set_index('PERIODO', inplace=True)
```

- **Reindexación con el rango de fechas completo**: Se asegura de que el DataFrame contenga todas las fechas posibles en el rango de tiempo, incluyendo aquellas sin datos. Esto es útil para manejar datos faltantes.

```
inicio = df.index.min()
fin = df.index.max()
date_range = pd.date_range(start=inicio, end=fin, freq='1H')
df = df.reindex(date_range)
```

2. Imputación de datos faltantes

- **Interpolación lineal**: Se utiliza una interpolación lineal para imputar (rellenar) hasta 5 valores consecutivos faltantes. Este método es útil para series temporales donde los datos cercanos tienen una relación fuerte.

```
df_copy = df.copy(deep=True)
df_copy.set_index('PERIODO', inplace=True)
df_copy = df_copy.interpolate(method='linear', limit=5,
limit_direction='forward', axis=0)
```

3. Pronóstico de datos horarios mediante Random Forest

3.1 Preparar los datos

- **Creación de características retrasadas**: Se generan características de la serie temporal creando variables de retardo (`lag`). Esto significa que se usan los valores de PM10 de horas anteriores como características para predecir el valor actual o futuro.

```
def create_lagged_features(series, n_lags):
    df_tmp = pd.DataFrame(series)
    for lag in range(1, n_lags + 1):
        df_tmp[f'lag_{lag}'] = df_tmp[tx_campo].shift(lag)
    df_tmp.dropna(inplace=True)
    return df_tmp

n_lags = 24  # Número de horas anteriores a considerar
df_lagged = create_lagged_features(df_copy[tx_campo], n_lags)
```

- **División de datos**: Los datos se dividen en características (`X`) y el objetivo (`y`), que son los valores que se intentan predecir.

```
x, y = df_lagged.drop(tx_campo, axis=1), df_lagged[tx_campo]
```

3.2 Entrenar el modelo Random Forest

- **Modelo de Random Forest**: Se entrena un modelo de `RandomForestRegressor` con los datos preparados. El modelo aprende la relación entre las características retrasadas y los valores actuales de PM10.

```
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(x, y)
```

3.3 Realizar predicciones multi-step recursivas

- **Función de predicción recursiva**: Esta función toma el modelo entrenado y predice valores futuros de PM10 de manera recursiva, es decir, usando las predicciones previas como nuevas entradas para predicciones adicionales.

```
def predict_recursive_from_start(model, df_copy, n_lags, steps):
    # Código para predecir usando el modelo entrenado
```

- **Generación de predicciones**: Se realizan predicciones para las próximas 24 horas y se combinan con los datos de entrenamiento para crear un DataFrame que contenga tanto los valores reales como los pronosticados.

```
predictions = predict_recursive_from_start(model, df_copy, n_lags, 24)
```

Visualización y guardado de resultados

- **Graficar las predicciones**: Se crea una gráfica que muestra tanto los valores reales de PM10 como las predicciones generadas por el modelo.

```
plt.figure(figsize=(12, 6))
plt.plot(df_copy.index, df_copy[tx_campo], label='Actual')
plt.plot(forecast.index, forecast['Prediccion'], label='Pronostico',
color='red')
plt.legend()
plt.xlabel(tx_campo_periodo)
plt.ylabel(tx_campo)
plt.title('Pronóstico de 24 Horas de datos Horarios de PM10 de Santa Anita -
2023 Aplicando Inteligencia Artificial')
plt.show()
```

- **Guardar resultados**: Finalmente, el DataFrame con las predicciones se guarda en un archivo CSV para análisis posterior.

```
lv_nom_file_predicciones= lv_nom_file + '_Predicciones_RF.csv'
data_forecast.to_csv(lv_nom_file_predicciones, sep=';', header=True,
index=True)
```

Resumen

Este programa utiliza Random Forest para predecir los niveles de PM10 en base a datos horarios pasados. Comienza cargando y limpiando los datos, imputa valores faltantes, entrena un modelo de Random Forest usando características de retardo, y luego utiliza ese modelo para hacer predicciones futuras. Los resultados se visualizan y se almacenan para su análisis.