Proyección horaria de la demanda de energía para un operador de red

Luis Carlos Cubides Rivera

Monografía presentada para optar al título de Especialista en Analítica y Ciencia de Datos

Tutor

Raul Ramos Pollan, Doctor (PhD)

Universidad de Antioquia

Facultad de Ingeniería

Especialización en Analítica y Ciencia de Datos

Medellín, Antioquia, Colombia

2021

Cita	(Muñoz Zapata & Martínez Naranjo, 2018)
Referencia **Estilo APA 7 (2020)**	Muñoz Zapata, L., & Martínez Naranjo, J. A. (2018). *Archivo fotográfico de la Universidad de Antioquia: valoración histórica de las fotografías, 1997 - 2003* [Trabajo de grado especialización]. Universidad de Antioquia, Medellín, Colombia.

Especialización en Analítica y Ciencia de Datos, Cohorte Seleccione cohorte posgrado.

Grupo de Investigación Seleccione grupo de investigación UdeA (A-Z)

Seleccione centro de investigación UdeA (A-Z).

UNIVERSIDAD DE ANTIOQUIA
Vicerrectoría de Docencia

Sistema de Bibliotecas

Elija un elemento.

Repositorio Institucional: http://bibliotecadigital.udea.edu.co

Universidad de Antioquia - www.udea.edu.co

Rector: Jonh Jairo Arboleda Cespedes.

Decano/Director: Jesus Francisco Vargas Bonilla.

Jefe departamento: Diana Margot Lopez Herrera.

Contenido

1. Resumen

La demanda de energía eléctrica es una variable fundamental para el crecimiento económico del país, ya que la electricidad es el motor que impulsa la economía y la revolución en la industria 4.0. Es una variable que tiene dependencia del crecimiento de la población, e industrial, se debe supervisar ya que para poder atenderla se requiere una generación que es lenta para entrar en línea, y costosa en el caso de que se requiera reservas que no se tenían proyectadas, lo anterior puede generar un apagón ocasionando sanciones para los operadores de red, pérdidas económicas para la industria, y en el peor de los afectos se puede ocasionar vidas en el sector salud. En este sentido, en esta monografía se propone el análisis y validación de diferentes modelos para la elaboración de la proyección de demanda hora a hora para un operador de red.

2. Descripción del problema

Los operadores de red son los encargados de suministrar y mantener con calidad el servicio de energía eléctrica hasta la puerta de todos los hogares, para Colombia son 30, y cada uno opera como un agente ante XM (compañía que experta del mercado eléctrico en Colombia), se encarga de operar y despachar a todos los agentes la energía que requiere para cumplir sus objetivos.

La planeación de sistemas eléctricos tiene como objetivo establecer las capacidades, reconfiguraciones y/o ampliaciones del sistema, además de definir el tiempo en que se deben realizar dichos cambios de manera óptima, de modo que el plan de expansión garantice el servicio de energía para los usuarios actuales y futuros con niveles de confiabilidad aceptables y con el menor costo posible. En la etapa de planeamiento de sistemas eléctricos se debe conocer de antemano las características de la demanda que debe suplir, es por este motivo que se hace importante realizar de manera previa un estudio que permita caracterizar la demanda de forma adecuada. En la operación de sistemas eléctricos, una tarea día a día, se debe predecir la demanda diaria para atender criterios regulatorios, prepararse para contingencias presentadas en el sistema, y conocer de antemano las necesidades de compra y venta en el mercado eléctrico.

2.1. Estado del arte

En respuesta a esta necesidad, múltiples metodologías han sido propuestas para determinar una caracterización de la demanda que permita dimensionar el sistema eléctrico sin incurrir en errores considerables que se traduzcan en un sistema eléctrico ineficiente o sobredimensionado. Dentro de las herramientas utilizadas para este propósito se pueden mencionar algunas como regresiones, sistemas inteligentes, series de tiempo y modelos econométricos y estadísticos.

Así como las herramientas utilizadas en el campo son variadas, las metodologías lo son también. Algunos trabajos utilizan el concepto de previsión espacial de demanda, en cuyo caso se pueden destacar trabajos como [1], [2] y [3], donde además de tener en cuenta factores tradicionales como temperatura y datos económicos, se estudia la ubicación y la magnitud de las futuras cargas teniendo en cuenta aspectos como el uso del suelo, sectorizando la demanda de acuerdo con la posibilidad de expansión que presenta la región.

En [4], el autor estudia y valida las variables que comúnmente se utilizan en los modelos de previsión de demanda a largo plazo, las cuales clasifica en tres grupos diferentes; esta clasificación se muestra en la Tabla 1 con algunos ejemplos de las variables que entran en cada una de estas clasificaciones. Dada la complejidad de los modelos de proyección de demanda la información ingresada al modelo

puede volverse redundante, por lo cual el autor argumenta que un modelo determinado no requiere del uso de todas las variables al mismo tiempo, y que además no existe una combinación óptima y única de variables. Debido a esto, se debe realizar un análisis de selección de variables de acuerdo con el modelo elegido, por ejemplo, en su trabajo, el autor aplica un método de análisis de regresión con el cual pretende obtener la mejor combinación de variables que modelan linealmente y con buen desempeño la demanda de energía y potencia eléctrica.

Tabla 1. Variables Relevantes en las metodologías de previsión de demanda a largo plazo

Macroeconómicas	**Características propias del sistema**	**Climatológicas**
• Producto Interno Bruto (PIB) • PIB per Cápita • Tamaño de la Población	• Precio de la Energía • Factor de Carga • Pérdidas del Sistema	• Temperatura Media • Velocidad del Viento • Humedad Relativa • Cantidad de Horas de Sol

En Colombia, La Unidad de Planeación Minero-Energética (UPME) realiza las proyecciones de la demanda de la energía eléctrica a largo plazo (anual) a partir de un modelo econométrico de combinación de pronósticos que emplea modelos multivariados como los VAR (Modelo de Vectores Autorregresivos) y los VEC (Modelo de Vectores de Corrección de Error) [5], utilizando las siguientes variables:

En [6] la UPME muestra proyecciones regionales de demanda de energía eléctrica y potencia máxima a nivel de Unidades de Control de Pronóstico (UCP) con un horizonte hasta el año 2030, estas proyecciones son desagregas para ocho (8) regiones del país (Centro, Costa-Caribe, Noroeste, Oriente, Valle, CQR, Tolima Grande y Sur), vale la pena destacar que para realizar estas proyecciones, UPME utiliza información macroeconómica regional aplicando métodos de series de tiempo y econométricos. Además, se menciona que para mejorar el desempeño de las proyecciones es necesario mantener el análisis de las tendencias de crecimiento, las periodicidades, el desarrollo económico de los sectores y los fenómenos climáticos de cada región.

Por otro lado, en [7], la UPME presenta un análisis en el cual se demuestra la correlación de la temperatura con respecto a la demanda eléctrica en Colombia, utilizando datos históricos de dichas variables de 1991 al 2013. Este análisis es realizado para tres (3) modelos de proyección de demanda: modelo VAR Exógeno, modelo VAR Endógeno y modelo de Series de Tiempo.

Dentro de las herramientas más utilizadas en la literatura especializada se pueden resaltar dos tipos: el primero, es una herramienta que utiliza procesos estadísticos el cual se conoce como el método estadístico y econométrico. El segundo método, enmarca técnicas computacionales que emulan sistemas biológicos reales y los utiliza para la toma de decisiones, los cuales son conocidos como sistemas inteligentes.

2.2. Justificación

- Resolución CREG 015 de 2018 exige que las proyecciones de demanda para los operadores de red estén entre un +- 5 % del valor real.

- El CND debe realizar proyecciones de demanda diaria para establecer las compras que debe realizar al mercado eléctrico colombiano, que generadores entran a operar y cuales quedan por respaldo. (Esto es una bolsa, si se pasa por encima del real, se debe pagar por la disponibilidad y maquinas rodantes que se tienen en el mercado, si se prevé por debajo, pueden ocurrir apagones a nivel nacional).

- En esta propuesta se quiere definir un modelo basado en históricos, que permita predecir el comportamiento en tiempo real de la demanda (hora a hora), de manera que se mantenga dentro de un +-3% de su margen real y no genere sobrecostos por tener que comprar una energía mas cara respecto a la que ya está en la bolsa de mercado eléctrico, o no utilizar energía que ya se había pactado mediante un contrato previamente definido con el agente generador.

3. Datos

La información alojada se descarga de XM (Entidad encargada del manejo del mercado de comercialización, operación y despacho de energía eléctrica en Colombia), posteriormente se realiza su conversión a un DataFrame para su posterior trabajo. la descarga se realiza del siguiente enlace: http://poítalbissís.xm.com.co/dmnd/Paginas/Histoíicos/Histoíicos.aspx se utiliza como información para el entreamiento los consumos horarios para el año 2020.

3.1. Datos Originales

Los datos originales contienen para cada operador de red la siguiente información y guarda la estructura que se presenta en la:

- Potencia: Es la proporción por unidad de tiempo en que fluye la energía eléctrica, se mide en kW (kilo Watts).
- Día
- Código del operador de red
- Hora

Tabla 1. Estructura de los datos originales

Fecha	Código OR	0	1	2	3	…	23
01/01/2020	OR1	Información de demanda para cada hora (kW)					
⋮							
31/12/2020	OR30						

3.2. Dataset y descriptiva

El archivo entregado es cargado en un DataFrame, se procede a renombrar las columnas y a partir de ahí comenzar a conocer los datos, en total se tienen 10340 filas y 26 columnas, como se presenta en la Figura 1.

```
[ ]  Demandas="https://raw.githubusercontent.com/cubides0905/TareaFinal/main/Demanda_por_OR_2020.xlsx"

     Data=pd.read_excel(Demandas,skiprows=range(0, 2))
```

Se procede a realizar el renombramiento de las columnas para el DataFrame

```
[ ]  Data.columns = ['Fecha','Codigo_OR','P_H_1','P_H_2','P_H_3','P_H_4','P_H_5','P_H_6','P_H_7','P_H_8','P_H_9','P_H_10','P_H_11','
```

En los Datos, se puede observar en la Columna Codigo_OR, que estan todos los operadores de red de Colombia, para la iteración inicial se va a trabajar con el operador de red EPM (Empresas Publicas de Medellin)

Figura 1. Cargue de los datos y renombramiento de columnas

Como paso inicial se selecciona el operador al que se le va a proyectar la demanda horario de energía, para este caso se selecciona EPMD (Empresas Públicas de Medellín), que suministra energía para todo el departamento de Antioquia, teniendo en total un dataset filtrado por 360 filas y 26 columnas, como se presenta en la Figura 2.

```
# datos solo para EPM
Data_EPM=Data.loc[(Data.Codigo_OR=='EPMD')]
```

```
[10] Data_EPM.head()
```

	Fecha	Codigo_OR	P_H_1	P_H_2	P_H_3	P_H_4	P_H_5	P_H_6	P_H_7	P_H_8	P_H_9	P_H_10	P_H_11	P_H_12	P_
21	2020-01-01	EPMD	771098.49	739418.44	697880.39	681451.32	666248.46	668138.23	630350.37	635025.00	668838.45	715133.57	760698.27	806726.13	8366
49	2020-01-02	EPMD	699244.83	661426.70	648763.48	651388.05	691238.95	753844.13	823729.39	919742.29	999973.37	1058154.81	1124511.33	1184467.68	12151
77	2020-01-03	EPMD	830583.01	786940.56	764928.63	759832.53	801235.69	860315.69	917512.04	1003990.03	1082409.82	1134224.85	1203232.55	1253679.04	12728
105	2020-01-04	EPMD	851369.34	808392.80	785544.74	783358.26	807509.66	846100.96	879332.51	959390.26	1037183.35	1098921.65	1167675.87	1209324.43	12211
133	2020-01-05	EPMD	841663.92	787890.51	763032.22	742671.24	747011.74	761651.84	746680.86	789569.65	849085.42	902067.58	953033.31	996818.49	10179

```
len(Data_EPM)
```

366

Figura 2. Filtrado de datos por OR = EPMD

Para conocer un poco la data, se procede a reflejar una curva promedio de demanda para el OR, encontrando un comportamiento como se presenta en la Figura 3. Se puede observar que se tienen diferentes picos, entre ellos a la hora 12 y hora 21, que es cuando se incrementa la producción industrial y consumo residencial, respectivamente.

```
DemandaPromedio=list(Data_EPM.mean())
plt.plot(np.linspace(0,24,24),DemandaPromedio,'b',label='DemandaPromedio')
plt.xlabel('Hora')
plt.ylabel('Potencia [Kw]')
plt.title(f'Curva de demanda promedio anual')
plt.legend()
plt.show()
```

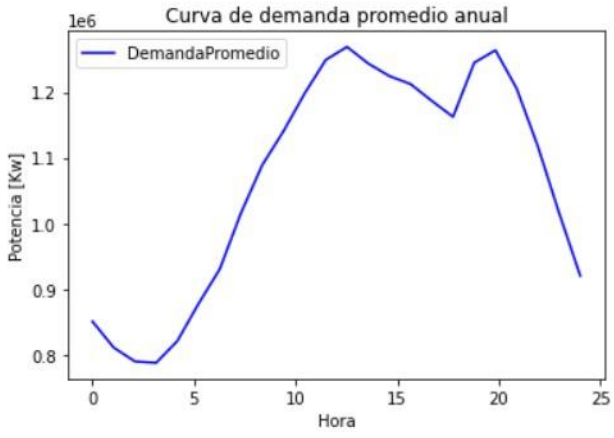

Figura 3. Caracterización de curva de demanda promedio

4. Proceso de analítica

En la sección de analítica, se procede a presentar como se abordó el problema expuesto en esta monografía. Inicialmente, un preprocesamiento de las variables e inclusión de algunas características que se pueden extraer de estos datos con base a la experiencia en el sector. Seguidamente, se presentan los diferentes modelos que se aplicaron para realizar la proyección.

4.1. Preprocesamiento

Hay algo que desde la experiencia en el sector eléctrico se ha obtenido, y es que el consumo de energía disminuye los fines de semana y festivos comparado con el transcurso de la semana, lo anterior es claro dado que los fines de semana algunas industrias no están trabajando. En este sentido, se procede a realizar la instalación de la librería holidays_co, que permite identificar por medio de la fecha que días son festivos. En este sentido, como se presenta en la Figura 4, se procede a detectar los días festivos y se marca con 1 los que lo son, adicionalmente, se enumera el día de la semana iniciando en 0 para lunes y finalizando en 6 para el domingo.

```
import holidays_co
Fechas=list(Data_EPM.Fecha)
DiaSemana=[]
Holiday=[]
for i in range(len(Fechas)):
  dt=Fechas[i]
  year, month, day = (int(x) for x in dt.split('-'))
  ans = datetime.date(year, month, day)
  if holidays_co.is_holiday_date(ans)==True:
    Holiday.append(1)
  else:
    Holiday.append(0)
  if ans.weekday()==6:
    Holiday[i]=2
  if ans.weekday()==5:
    Holiday[i]=3
  DiaSemana.append(ans.weekday()) # 0:lunes, 1:Martes, 2: Miercoles, 3:Jueves, 4 : viernes, 5 : Sabado, 6 Domingo
```

```
[ ]  Data_EPM.insert(1,'Day',DiaSemana,True)
     Data_EPM.insert(2,'Holiday',Holiday,True)
```

```
[ ] Data_EPM
```

	Fecha	Day	Holiday	Codigo_OR	P_H_1	P_H_2	P_H_3	P_H_4	P_H_5	P_H_6	P_H_7	P_H_8
21	2020-01-01	2	1	EPMD	771098.49	739418.44	697880.39	681451.32	666248.46	668138.23	630350.37	635025.00
49	2020-01-02	3	0	EPMD	699244.83	661426.70	648763.48	651388.05	691238.95	753844.13	823729.39	919742.29

Figura 4. Determinación de Festivos y días de la semana en función de la fecha

En la Figura 5, se presenta la caracterización de la curva de demanda para cada tipo de día, después de realizar el procedimiento anteriormente mencionado, se procede a extraer el valor promedio para todos los tipos de día del año. Como se puede notar, el dataset cumple lo que se había mencionado anteriormente en función de la experiencia del sector eléctrico, los días festivos y domingos son los que menos demanda tienen, adicionalmente, para el día Jueves para ser el día más representativo para el año 2020.

```
plt.plot(np.linspace(0,24,24),Curva_Demanda_Festivo,'b',label='Festivo')
plt.plot(np.linspace(0,24,24),Curva_Demanda_Lunes,'g',label='Lunes')
plt.plot(np.linspace(0,24,24),Curva_Demanda_Martes,'r',label='Martes')
plt.plot(np.linspace(0,24,24),Curva_Demanda_Miercoles,'c',label='Miercoles')
plt.plot(np.linspace(0,24,24),Curva_Demanda_Jueves,'m',label='Jueves')
plt.plot(np.linspace(0,24,24),Curva_Demanda_Viernes,'y',label='Viernes')
plt.plot(np.linspace(0,24,24),Curva_Demanda_Sabado,'k',label='Sabado')
plt.plot(np.linspace(0,24,24),Curva_Demanda_Domingo,'g--',label='Domingo')
plt.xlabel('Hora')
plt.ylabel('Potencia [Kw]')
plt.title(f'Curva de demanda por tipo de día')
plt.legend()
plt.show()
```

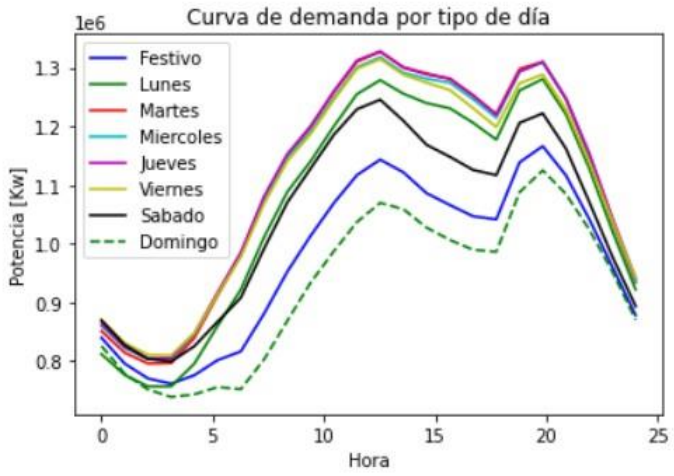

Figura 5.Caracterización de la curva de demanda para cada tipo de día

4.2. Partición de los datos para entrenamiento y prueba

Dado que lo que se quiere predecir es demanda, y el comportamiento obedece a una serie de tiempo, el particionamiento de los datos se debe realizar teniendo en cuenta la variable tiempo como se presenta en la Figura 6. Quiere decir, si se va a probar con una hora (i) los datos con los que se entrenan deben de ser una fecha anterior al día a proyectar (díaSel).

```
Train=DataModificada(Data_EPM,i+1,4) # datos de entrenamiento
X2 = Train.iloc[:,0:Train.shape[1]-1] # entradas
y2 = Train.iloc[:,Train.shape[1]-1:Train.shape[1]] # salida
diaSel=len(Train)/2+ np.round(rnd.random()*len(Train)/2) # dia seleccionado a proyectar
X_train2=X2.iloc[0:int(diaSel)-1,:] # datos para entrenar
y_train2=y2.iloc[0:int(diaSel)-1,:] # salidas para entrenar
X_test2=X2.iloc[int(diaSel)-1:int(diaSel),:] # datos para probar
y_test2=y2.iloc[int(diaSel)-1:int(diaSel),:] # salida para probar
```

Figura 6. Líneas de código que realiza el particionamiento de los datos

Adicionalmente, en cantidad de horas de entrada, se usa el parámetro (i), para establecer con cuantas horas de anterioridad se van a ingresar los datos. Quiere decir, si i=5, y la hora a proyectar es 10, se selecciona desde las 5 de la mañana de todos los días los datos de entrenamiento.

4.3. Modelos Propuestos

Para realizar la proyección de la hora i, se procede a definir la implementación de diferentes modelos para comparar su desempeño, se implementan de vaseline, de regresión línea, máquinas de soporte vectorial, árboles de decisión y random forest. Para el caso del vaseline, se utiliza el criterio de que la demanda se mantenga constante respecto a la hora anterior. En las Figura 7 a Figura 11 se presenta cada uno de los códigos desarrollados para los modelos propuestos.

```
ResultadoVAS=[]
for i in range(24):
  Train=DataModificada(Data_EPM,i+1,4)
  X2 = Train.iloc[:,0:Train.shape[1]-1]
  y2 = Train.iloc[:,Train.shape[1]-1:Train.shape[1]]
  diaSel=len(Train)/2+ np.round(rnd.random()*len(Train)/2)
  X_train2=X2.iloc[0:int(diaSel)-1,:]
  y_train2=y2.iloc[0:int(diaSel)-1,:]
  X_test2=X2.iloc[int(diaSel)-1:int(diaSel),:]
  y_test2=y2.iloc[int(diaSel)-1:int(diaSel),:]
  test_predictions = X_test2.iloc[0,5]
  test_error = custom_error(np.array(list(y_test2.iloc[:,0])), test_predictions)
  ResultadoVAS.append(test_error)
```

Figura 7. Código desarrollado para el modelo Vaseline

```
Resultado=[]
for i in range(24):
  Train=DataModificada(Data_EPM,i+1,4) # datos de entrenamiento
  X2 = Train.iloc[:,0:Train.shape[1]-1] # entradas
  y2 = Train.iloc[:,Train.shape[1]-1:Train.shape[1]] # salida
  diaSel=len(Train)/2+ np.round(rnd.random()*len(Train)/2) # dia seleccionado a proyectar
  X_train2=X2.iloc[0:int(diaSel)-1,:] # datos para entrenar
  y_train2=y2.iloc[0:int(diaSel)-1,:] # salidas para entrenar
  X_test2=X2.iloc[int(diaSel)-1:int(diaSel),:] # datos para probar
  y_test2=y2.iloc[int(diaSel)-1:int(diaSel),:] # salida para probar
  lr=LinearRegression()
  lr.fit(X_train2,y_train2)
  evaluation = evaluate_model(
      model=lr,
      metric=custom_error2,
      X_train=X_train2,
      y_train=y_train2,
      X_test=X_test2,
      y_test=y_test2
  )
  Resultado.append(evaluation['test_error'])
```

Figura 8. Código desarrollado para el modelo de regresión lineal

```
[ ]  from sklearn.svm import SVR
     svr=SVR(kernel='linear',C=0.1,epsilon=0.001)
```

```
[ ]  # se pone un valor para C de 0.3  kernel= linear
     ResultadoSVR=[]
     for i in range(24):
       Train=DataModificada(Data_EPM,i+1,4)
       X2 = Train.iloc[:,0:Train.shape[1]-1]
       y2 = Train.iloc[:,Train.shape[1]-1:Train.shape[1]]
       diaSel=len(Train)/2+ np.round(rnd.random()*len(Train)/2)
       X_train2=X2.iloc[0:int(diaSel)-1,:]
       y_train2=y2.iloc[0:int(diaSel)-1,:]
       X_test2=X2.iloc[int(diaSel)-1:int(diaSel),:]
       y_test2=y2.iloc[int(diaSel)-1:int(diaSel),:]

       svr.fit(X_train2,y_train2)
       train_predictions = svr.predict(X_train2)
       test_predictions = svr.predict(X_test2)
       train_error = custom_error(np.array(list(y_train2.iloc[:,0])), train_predictions)
       test_error = custom_error(np.array(list(y_test2.iloc[:,0])), test_predictions)
       ResultadoSVR.append(test_error)
```

Figura 9. Código desarrollado para el modelo de máquinas de soporte vectorial

```
[ ]  from sklearn.tree import DecisionTreeRegressor
     adr=DecisionTreeRegressor(max_depth=20)
```

```
     ResultadoADR=[]
     for i in range(24):
       Train=DataModificada(Data_EPM,i+1,4)
       X2 = Train.iloc[:,0:Train.shape[1]-1]
       y2 = Train.iloc[:,Train.shape[1]-1:Train.shape[1]]
       diaSel=len(Train)/2+ np.round(rnd.random()*len(Train)/2)
       X_train2=X2.iloc[0:int(diaSel)-1,:]
       y_train2=y2.iloc[0:int(diaSel)-1,:]
       X_test2=X2.iloc[int(diaSel)-1:int(diaSel),:]
       y_test2=y2.iloc[int(diaSel)-1:int(diaSel),:]

       adr.fit(X_train2,y_train2)
       train_predictions = adr.predict(X_train2)
       test_predictions = adr.predict(X_test2)
       train_error = custom_error(np.array(list(y_train2.iloc[:,0])), train_predictions)
       test_error = custom_error(np.array(list(y_test2.iloc[:,0])), test_predictions)
       ResultadoADR.append(test_error)
```

Figura 10. Código desarrollado para el modelo de árboles de decisión

```
from sklearn.ensemble import RandomForestRegressor
RFR=RandomForestRegressor(n_estimators=300,max_depth=15)
```

```
ResultadoRFR=[]
for i in range(24):
  Train=DataModificada(Data_EPM,i+1,4)
  X2 = Train.iloc[:,0:Train.shape[1]-1]
  y2 = Train.iloc[:,Train.shape[1]-1:Train.shape[1]]
  diaSel=len(Train)/2+ np.round(rnd.random()*len(Train)/2)
  X_train2=X2.iloc[0:int(diaSel)-1,:]
  y_train2=y2.iloc[0:int(diaSel)-1,:]
  X_test2=X2.iloc[int(diaSel)-1:int(diaSel),:]
  y_test2=y2.iloc[int(diaSel)-1:int(diaSel),:]
  RFR.fit(X_train2,y_train2)
  train_predictions = RFR.predict(X_train2)
  test_predictions = RFR.predict(X_test2)
  train_error = custom_error(np.array(list(y_train2.iloc[:,0])), train_predictions)
  test_error = custom_error(np.array(list(y_test2.iloc[:,0])), test_predictions)
  ResultadoRFR.append(test_error)
```

Figura 11. Código desarrollado para el modelo de Random Forest

Los anteriores modelos descritos, son entrenados utilizando el dataset de entrada como se describe en la sección 4.2. Un modelo adicional que se pueba es ingresando los datos de potencia como una serie de tiempo y realizar la respectiva proyección con redes neuronales recurrentes. Se utiliza el modelo de red neuronal que se presenta en la Figura 12. El código utilizado se presenta en la

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 5)                 15

 dense_5 (Dense)             (None, 1)                 6

=================================================================
Total params: 21
Trainable params: 21
Non-trainable params: 0
_________________________________________________________________
```

Figura 12. Modelo de red neuronal recurrente para proyección de serie de tiempo de potencia

```
ErrorRed=[]
for i in range(24):
  train_size = len(Acumulado)-24-i
  test_size = len(Acumulado) - train_size
  train, test = np.array(Acumulado[0:train_size]), np.array(Acumulado[train_size:len(Acumulado)])

  # normalize the dataset
  scaler = MinMaxScaler(feature_range=(0, 1))
  trainN = scaler.fit_transform(train)
  testN = scaler.transform(test)

  #tensor formating

  X_train, y_train = split_sequence(trainN, look_back)
  X_test, y_test = split_sequence(testN, look_back)
  #print('Train',X_train[:10])
  #print('Test',y_train[:10])

  model1.compile(optimizer='adam',loss='mse')
  model1.fit(X_train.reshape(X_train.shape[0],look_back),y_train.flatten(),epochs=200, verbose=0)
  trainPredict, testPredict = EstimaRMSE(model1,X_train,X_test,y_train,y_test,scaler,look_back)
  #PintaResultado(Acumulado,trainPredict,testPredict,look_back)
  testY = scaler.inverse_transform([y_test.flatten()])
  ErrorRed.append(np.abs(testY[0][0]-trainPredict[0][0])*100/testY[0][0])
```

Figura 13. Código desarrollado para el modelo de red neuronal recurrente

5. Evaluación de modelos

Para evaluar los modelos se utiliza el siguiente error, la implementación se presenta en la Figura 14.

$$Error = \max \frac{(Y_{true} - Y_{pred})}{Y_{true}}$$

```
def custom_error(
    y_true,
    y_pred
 ):

    diff = (y_true - y_pred)/y_true

    return max(np.abs(diff)*100)
```

Figura 14. Implementación del error

Se realiza la evaluación de los modelos propuestos en la sección 4, encontrando el comportamiento que se presenta en cada una de las siguientes gráficas.

Figura 15. Comportamiento del error en las 24 horas para el modelo vaseline

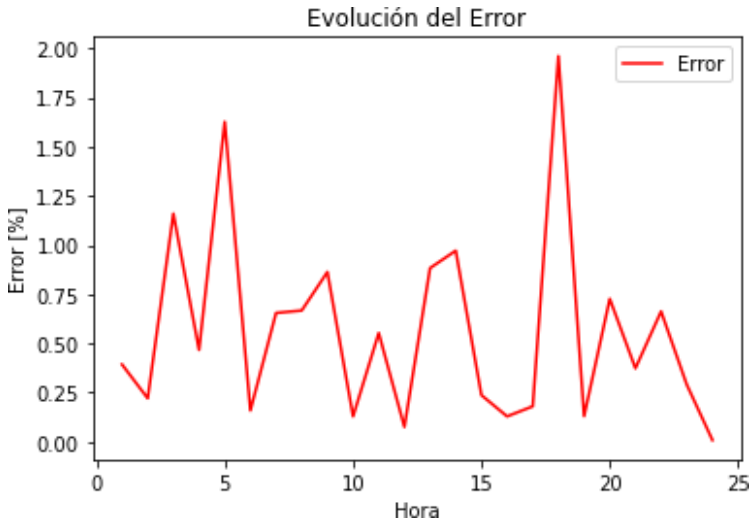

Figura 16. Comportamiento del error en las 24 horas para el modelo de Regresión lineal

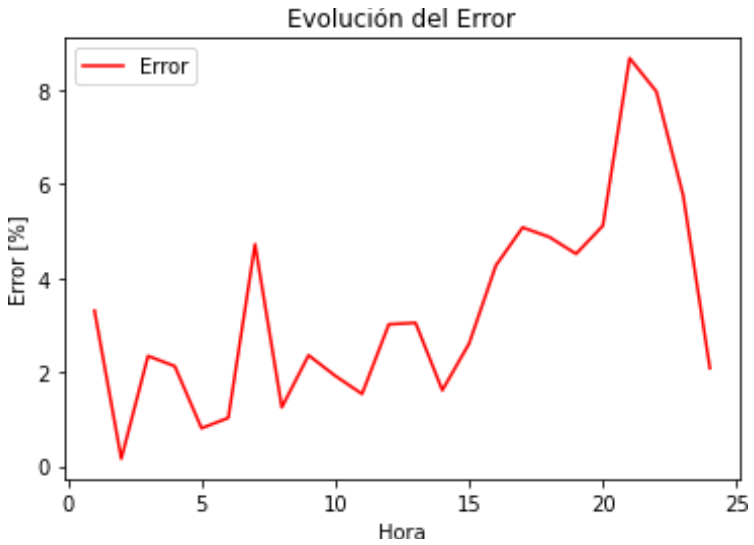

Figura 17. Comportamiento del error en las 24 horas para el modelo de Maquina soporte vectorial

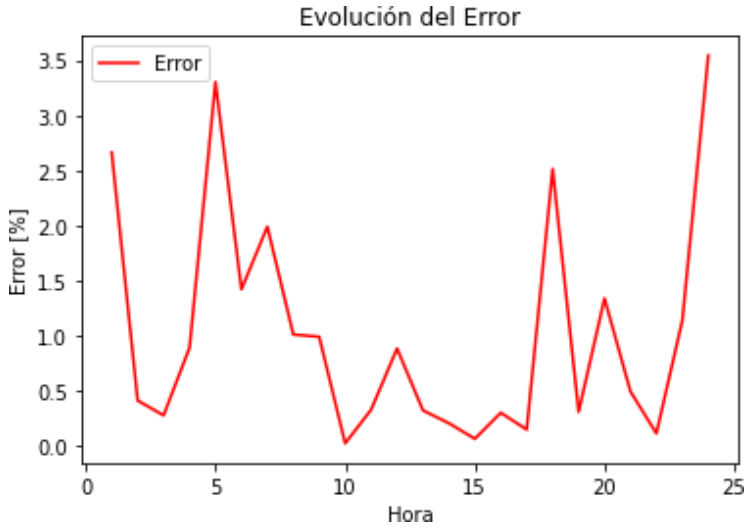

Figura 18. Comportamiento del error en las 24 horas para el modelo de Arboles de decisión

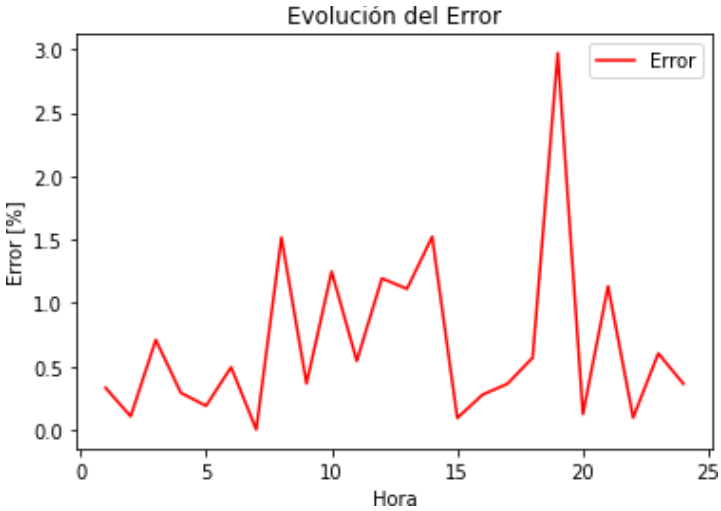

Figura 19. Comportamiento del error en las 24 horas para el modelo de Random Forest

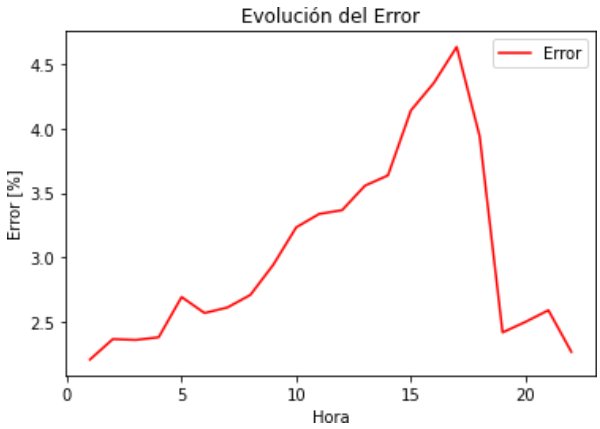

Figura 20. Comportamiento del error en las 24 horas para el modelo de Red neuronal recurrente

A continuación, se presenta en la tabla la ejecución de todos los modelos y la comparación de acuerdo con los errores promedio obtenidos para todas las 24 horas de ejecución. Se puede observar que 4 de los 6 modelos propuestos tienen un error promedio menor a 3%.

Tabla 2. Comparación de modelos propuestos

Modelo	**Promedio Error (%)**
Vaseline	4.66
Lineal	0.87
Arbol de decisión	075
Random Forest	0.73
Máquinas de soporte vectorial	3.88
Red Neuronal Recurrente	3.03

Adicionalmente, se realiza una búsqueda iterativa de la cantidad de datos que se debe ingresar al modelo lineal para que el error se minimice, observando en la

Figura 21 que a medida que se agrega más datos históricos para el entramiento, se puede obtener un menor error.

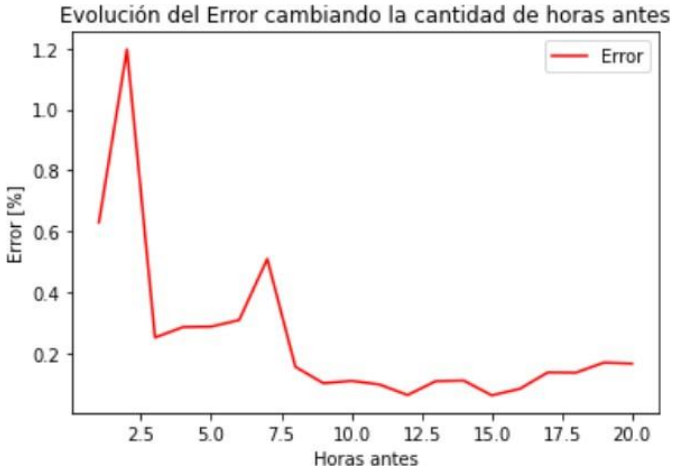

Figura 21. Evolución del modelo al agregar mayor historial de horas

6. Despliegue del modelo en AZURE

El modelo es desplegado en Azure como lo presenta la estructura de la Figura 22, y realizando los siguientes pasos:

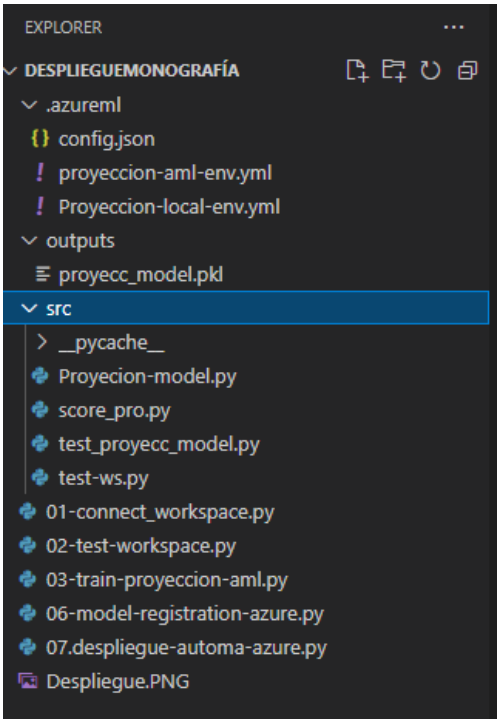

Figura 22. Estructura de modelo para despliegue en azure

1. Creación del workspace

La creación del workspace, ermite generar el espacio de trabajo y conexión entre el visual studio code y la plataforma de Azure, se utiliza una máquina virtual ESTÁNDAR_D2_V3 ((2 núcleos, 8 GB de RAM y disco de 50 GB), como lo presenta la Figura 23,

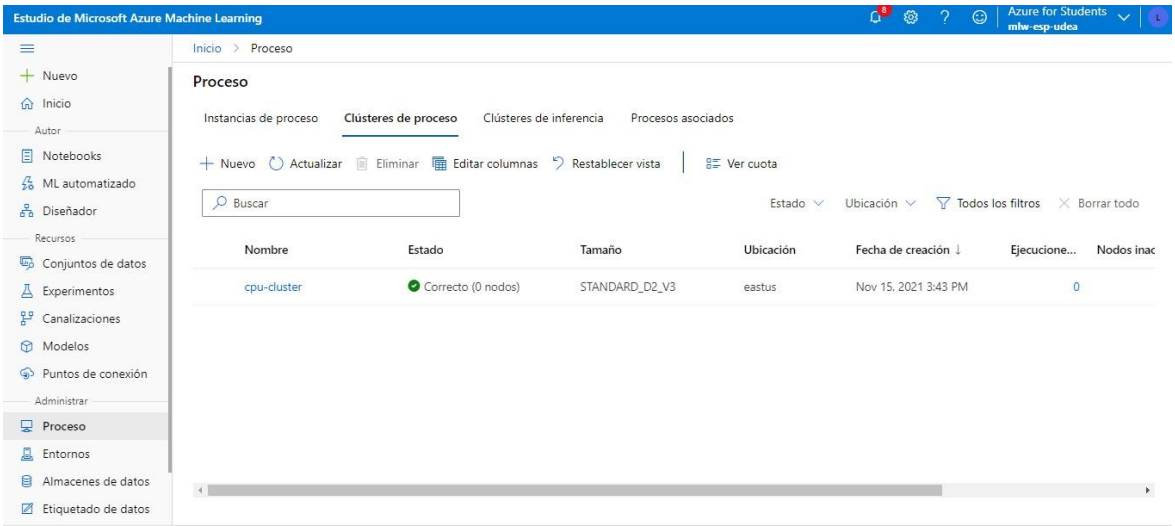

Figura 23. Creación del workspace

2. Entreamiento y pruebas del modelo

El entrenamiento del modelo se realiza para la hora a proyectar igual a 1, quiere decir se van a ingresar el valor de potencia para las 4 horas anteriores, adicionalmente el día de la semana y si es

festivo o no, el modelo se encuentra en el archivo “Proyecion-model.py”, y se selecciona para realizar las proyecciones una regresión lineal.

Al ejecutar el entramiento en un ambiente local **Proyeccion-local-env,** el error que se obtiene para definir el modelo **proyecc_model.pkl** es de 1.95%, como lo presenta la terminal en la Figura 24.

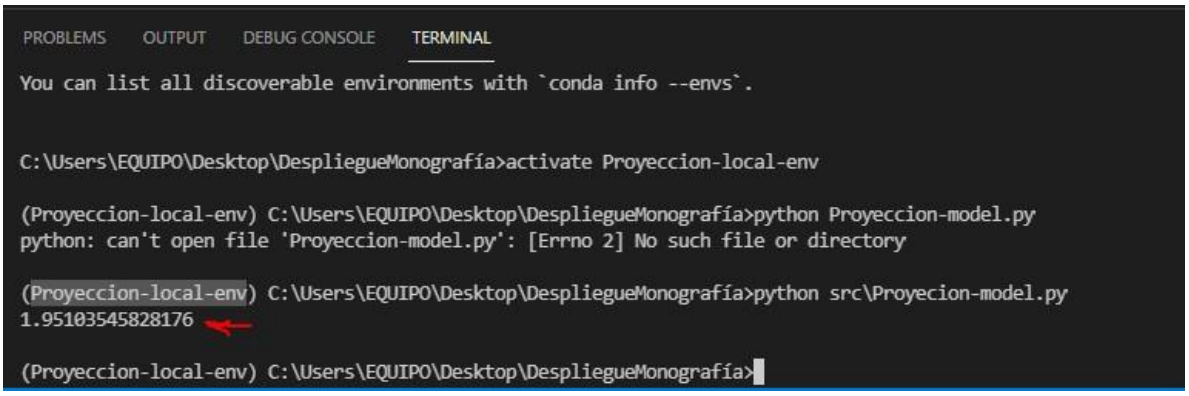

Figura 24.Entrenamiento de modelo

Posteriormente, mediante el código “test_proyecc_model.py”, se ejecuta de manera local las pruebas del modelo creado **proyecc_model.pkl,** encontrando un valor de proyección de **875284 Kw** para la potencia como lo presenta la Figura 25**,** un valor que de acuerdo con históricos está correcto ya que la hora 1 tiene a ser menor que la 24.

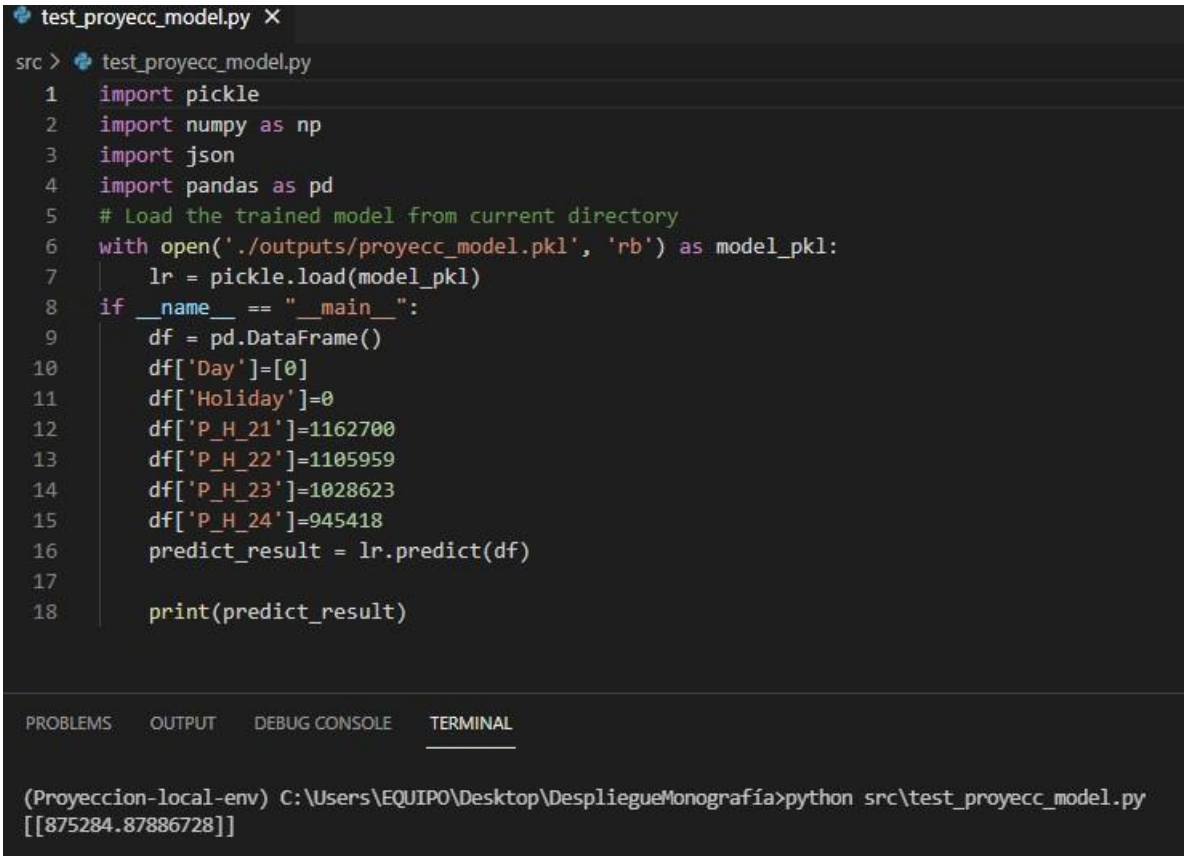

Figura 25. Pruebas de modelo creado

Finalmente, en el ambiente de azure **proyeccion-aml-env,** se procede a ejecutar los respectivos experimentos del modelo, se realiza 3 repeticiones hasta que se puede organizar un problema que se tenia con la instalación de una librería. La tarea se nombra como **Final-proyec-train-proyeccion.** Se puede observar en la Figura 26 que después de la primera ejecución los tiempos disminuyen, ya que se tiene el espejo de los datos en el azure.

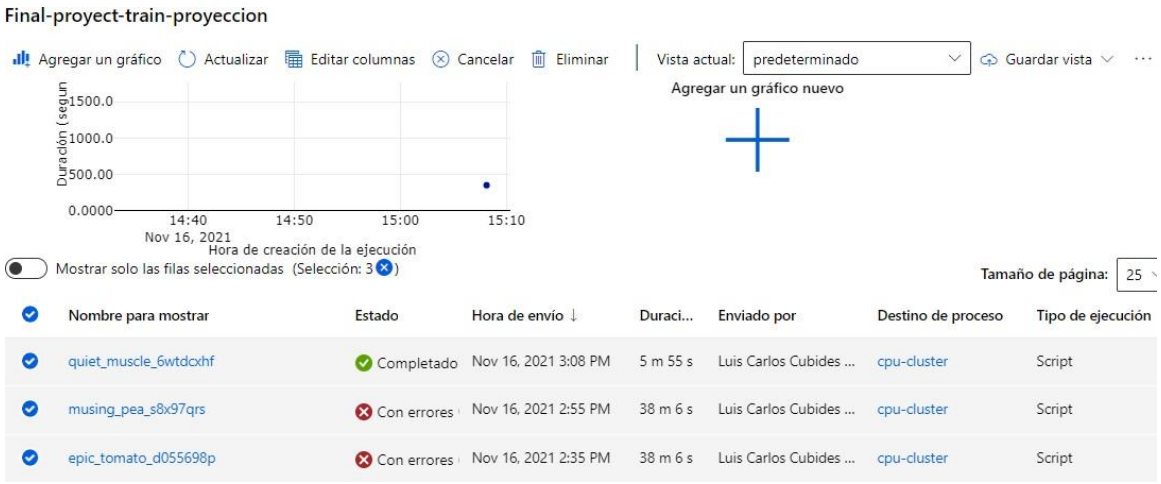

Figura 26.Experimentos implementados

3. Registro del modelo

El registro del modelo se realiza con el nombre **proyecc_model**, utilizando la estructura presentada en el código “model-registration-azure.py” y la Figura 27.

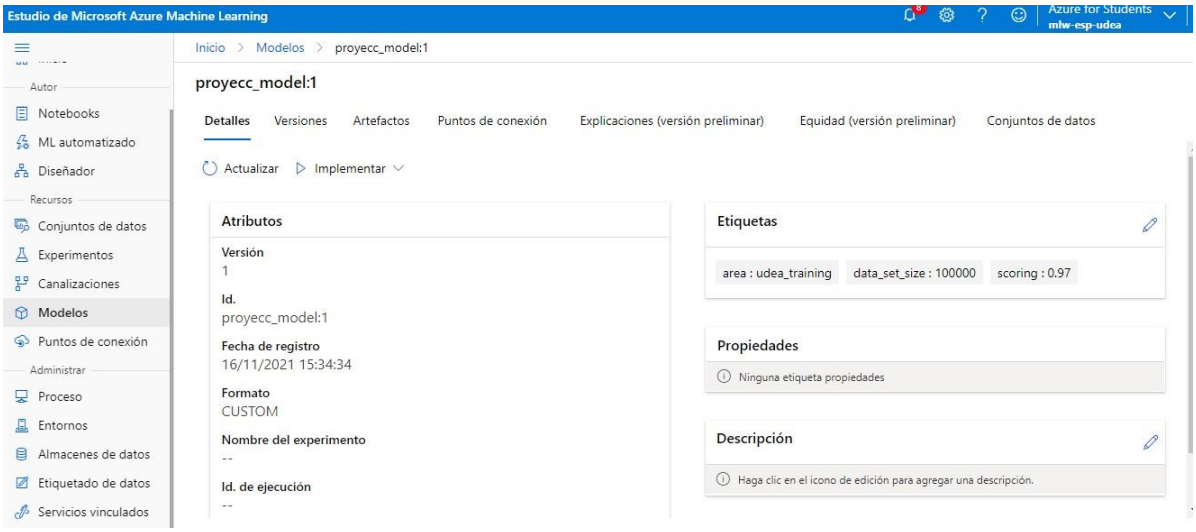

Figura 27. Registro del modelo

4. Despliegue del modelo en Azure

Utilizando la estructura presentada en el código “07.despliegue-automa-azure.py”, se genera el despliegue para el modelo creado **proyecc_model.pkl** mediante el servicio **proyec-model-service.** En la Figura 28 se presenta el despliegue del modelo, adicionalmente en la Figura 29 se procede a

ejecutar el web service generado utilizando el programa postman, la estructura de entrada es la siguiente:


```
{
"Days": 0,
"Holiday": 0,
"P_H_21":1162700,
"P_H_22":1102700,
"P_H_23":1002700,
"P_H_24":952700
}
```


Donde:
Days: Representa el número de día de la semana
Holidays: Si es festivo 1 o 0 en caso contrario.
P_H_21 a P_H_24: La potencia para cada una de las 4 horas anteriores a la hora a proyectar (1).

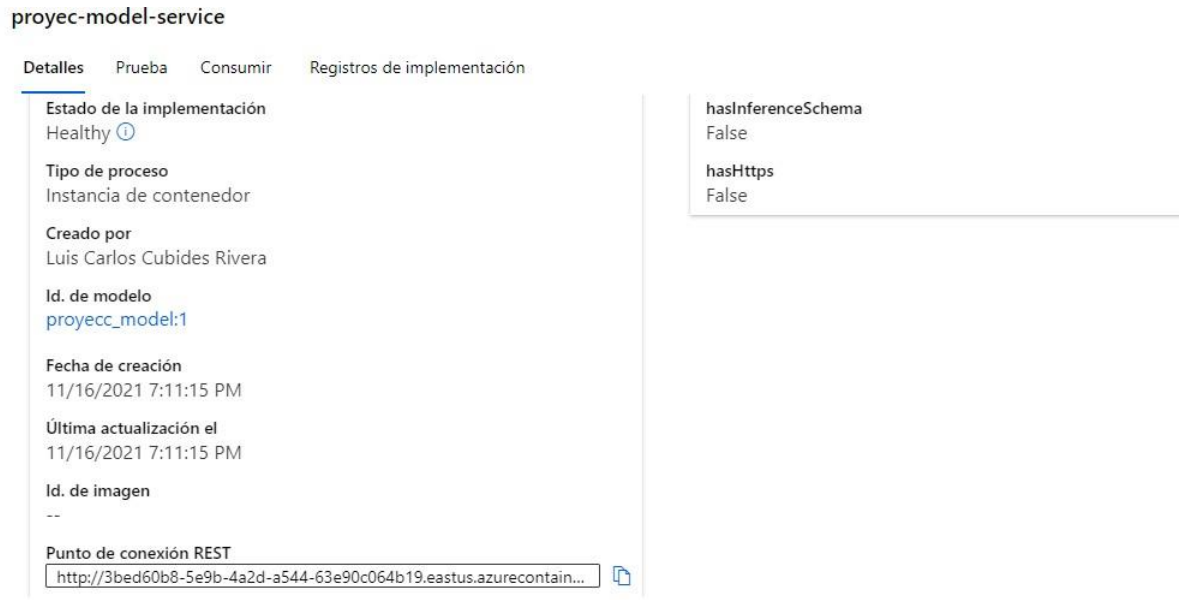

Figura 28. Despliegue de modelo en Azure

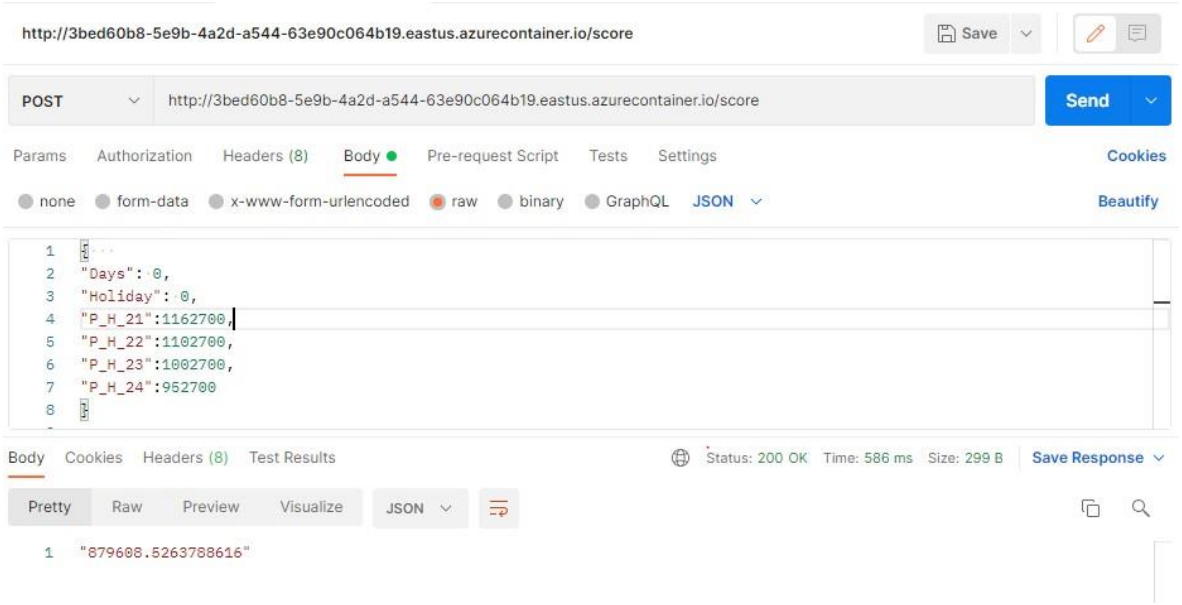

Figura 29. Aprovechamiento del servicio web utilizando postman

7. Conclusiones

- Mediante 3 de las 6 técnicas propuestas se logra resolver el problema presentado en esta monografía.
- El manejo de los datos, el conocimiento de este y su procesamiento generan gran valor en la proyección que se requiere realizar. Se puede observar que las Redes neuronales recurrentes a pesar de ser la técnica mas popular utilizada en la proyección de energía y demanda, en este documento como no se le dio un preprocesamiento anterior a los datos y se agregó como una serie de tiempo la potencia sin otras variables, fueron mejores las proyecciónes realizadas con árboles de decisión ya que si tenían la sensibilidad del día y si era festivo o no.
- En esta monografía se presenta el ciclo completo para la solución del problema, desde el manejo de los datos hasta el despliegue en la plataforma AZURE.

8. Bibliografía

[1] E. M. Carreño Franco, Previsão especial de Demanda em Sistemas de Distribuição com uma Base Reduzida de Dados, Ilha Solteira, 2008.

[2] J. D. Melo Trujillo, Aplicações de Sistemas Multiagentes na Previsão espacial de Demanda Elétrica em Sistemas de Distribuição, Ilha Solteira, 2010.

[3] E. M. Carreño y J. D. Melo, «Spatial load forecasting using a demand propagation approach,» de IEEE/PES Transmission and Distribution Conference and Exposition: Latin America, 2010.

[4] A. F. Palma LLewellyn, Pronóstico De Demanda De Energía Y Potencia Eléctrica En El Largo Plazo Para La Red De Chilectra S.A. Utilizando Técnicas De Minería De Datos, Santiago, 2015.

[5] UPME, Proyección de Demanda de Energía Eléctrica en Colombia, Bogotá, 2014.

[6] UPME, «Proyección Regional de Demanda de Energía Eléctrica y Potencia Máxima en Colombia,» Bogotá, Julio de 2016.

[7] UPME, «Relación entre Demanda de Energía Eléctrica y Temperatura en Colombia 1991-2013,» Bogotá, 2014.