UNIVERSIDAD LA SALLE

CARRERA DE INGENIERÍA DE SISTEMAS

REGRESIÓN LOGÍSTICA PARA DETERMINAR EL CLIMA DE UN ÁREA URBANA

LA PAZ – BOLIVIA

2020

Índice

Regresión Logística para determinar el clima de un área urbana

MARCO REFERENCIAL

Resumen

La simulación dinámica a largo plazo analiza el impacto de los cambios en las condiciones ambientales. El marco no convencional de la estructura del proceso programable aplicado apoya la realización de funciones interdisciplinarias en un marco transparente. El modelo de complejidad media desarrollado cubre las interacciones entre los procesos ecológicos y ambientales por lo que con los datos se busca predecir el clima en un área urbana con el modelo de regresión logística.

Introducción

En el presente documento se explora el estudio de la meteorología sobre cómo los datos atmosféricos y los cambios del clima, pueden utilizarse como factores ambientales, para predecir diferentes climas que enfrentan las personas a través del mundo.

Específicamente en las áreas urbanas, donde existe poca certeza acerca del clima que puede darse. Tomando en cuenta el área de Szeged(Hungría), se encuentran dificultades para predecir qué tipo de clima habrá, lo que a su vez dificulta la obtención de resultados aceptables debido a los altos cambios temporales y ambientales.

Además, desde un punto de vista científico, también se considera la importancia de caracterizar los fenómenos naturales, para evitar incidentes naturales. Las precipitaciones como la lluvia y la nieve, son fenómenos meteorológicos, que han sido ampliamente estudiados por su impacto en la economía y la industria.

Por ello gracias al avance de la tecnología y de la aplicación formal de la regresión logística se busca utilizar dicho modelo de predicción para enfatizar la relación estadística entre dos variables continuas conocidas como variables de predicción y variables de respuesta, para obtener la predicción de dos climas comunes en el área de Szeged, Hungría.

Descripción del problema

El archivo “**weatherHistory.csv”** es un conjunto de datos(dataset) con información sobre el clima de Szeged(Hungría), la cual, contiene datos meteorológicos de dicho lugar entre los años 2006 y 2016. El dataset posee una recopilación diaria, hora tras

hora, con características climatológicas como la temperatura normal(en grados centígrados), temperatura aparente(en grados centígrados), humedad, velocidad del viento(en kilómetros por hora), dirección del viento(en grados), visibilidad(en kilómetros), presión(en milibares), entre otros.

Con estos datos, conociendo el cálculo de estos valores como indicadores, se puede utilizar la regresión logística simple como un método de regresión para poder estimar la probabilidad de una variable cualitativa binaria, si hubo lluvia o nieve en la ciudad mencionada en función de una o varias variables cuantitativas.

Problema

Se necesita saber la tendencia del clima en los últimos diez años en la ciudad húngara de Szeged como parte del experimento. Medición que se puede usar para predecir nuevos resultados.

Objetivo

Determinar la relación matemática entre tipo de precipitación con datos meteorológicos de temperatura, velocidad, dirección del viento y presión

Objetivos específicos

- Dibujar un modelo de regresión logística usando variables cuantitativas.
- Utilizar el modelo para predecir el tipo de precipitaciones al momento de dar valores de entrada a la computadora.
- Obtener un coeficiente de predicción alto que nos permita aplicar el método.
- Calcular la relación entre las variables mediante una gráfica que permitirá saber si existe una tendencia marcada o lineal.

MARCO TEÓRICO

Regresión logística

La Regresión Logística Simple, desarrollada por David Cox en 1958, es un método de regresión que permite estimar la probabilidad de una variable cualitativa binaria en función de una variable cuantitativa.

Una de las principales aplicaciones de la regresión logística es la de clasificación binaria, en el que las observaciones se clasifican en un grupo u otro dependiendo del valor que tome la variable empleada como predictor. Por ejemplo, clasificar a un individuo desconocido como hombre o mujer en función del tamaño de la mandíbula.

Es importante tener en cuenta que, aunque la regresión logística permite clasificar, se trata de un modelo de regresión que modela el logaritmo de la probabilidad de pertenecer a cada grupo. La asignación final se hace en función de las probabilidades predichas.

Para evitar estos problemas, la regresión logística transforma el valor devuelto por la regresión lineal $(\beta 0 + \beta 1 \bullet X)$ empleando una función cuyo resultado está siempre comprendido entre 0 y 1. Existen varias funciones que cumplen esta descripción, una de las más utilizadas es la función logística (también conocida como función sigmoide):

$$\textit{función sigmoide: } \sigma(x) = \frac{1}{1 + e - x(1)}$$

Para valores de x muy grandes positivos, el valor de e−x es aproximadamente 0 por lo que el valor de la función sigmoide es 1. Para valores de x muy grandes negativos, el valor e−x tiende a infinito por lo que el valor de la función sigmoide es 0. Sustituyendo la x de la ecuación 1 por la función lineal (β0+β1X) se obtiene que:

$$\begin{aligned} P(Y=k|X=x) &= 11+e-(\beta 0+\beta 1X) \\ &= 1e\beta 0+\beta 1Xe\beta 0+\beta 1X+1e\beta 0+\beta 1X \\ &= 11+e\beta 0+\beta 1Xe\beta 0+\beta 1X \\ &= e\beta 0+\beta 1X1+e\beta 0+\beta 1X \end{aligned}$$

donde Pr(Y=k|X=x) puede interpretarse como la probabilidad de que la variable cualitativa Y, adquiera el valor k (el nivel de referencia, codificado como 1), dado que el predictor X tiene el valor x.

Servidor Jupyter para Visual Code

Jupyter Notebook es una aplicación cliente-servidor lanzada en 2015 por la organización sin ánimo de lucro Proyecto Jupyter. Permite crear y compartir documentos web en formato JSON que siguen un esquema versionado y una lista ordenada de celdas de entrada y de salida.

Estas celdas albergan, entre otras cosas, código, texto (en formato Markdown), fórmulas matemáticas y ecuaciones, o también contenido multimedia (Rich Media). El programa se ejecuta desde la aplicación web cliente que funciona en cualquier navegador estándar.

El requisito previo es instalar y ejecutar en el sistema el servidor Jupyter Notebook. Los documentos creados en Jupyter pueden exportarse, entre otros formatos, a HTML, PDF, Markdown o Python y también pueden compartirse con otros usuarios por correo electrónico, utilizando Dropbox o GitHub o mediante el visor integrado de Jupyter Notebook.

Recopilación y selección de datos

Para poder elaborar el presente proyecto se usó un conjunto de datos llamado **“weatherHistory.csv”** que se obtuvo de la página **Kaggle**, El conjunto de datos contiene el clima histórico alrededor de Szeged, Hungría - de 2006 a 2016 donde

tenemos 96,454 filas de datos y se tiene las siguientes columnas(se redujeron a 20,000 filas):

Fecha formateada, resumen, tipo de precipitación, temperatura (C), temperatura aparente (C), humedad, velocidad del viento (km / h), rumbo del viento (grados), visibilidad (km), cobertura ruidosa, presión (milibares), resumen diario.

Para su desarrollo no se utilizo todas las columnas del conjunto de datos, entre ellas fueron 'Tipo de precipitación', 'Temperatura (C)', 'Temperatura aparente (C)', 'Humedad', 'Velocidad viento (km/h)', 'Dirección del viento (grados)', 'Visibilidad (km)', 'Presión (milibares)'.

	Tipo de precipitacion	Temperatura (C)	Temperatura aparente (C)	Humedad	Velocidad viento (km/h)	Direccion del viento (grados)	Visibilidad (km)	Presion (millibares)
0	lluvia	9.472	7.389	0.89	14.120	251	15.826	1015.13
1	lluvia	9.356	7.228	0.86	14.265	259	15.826	1015.63
2	lluvia	9.378	9.378	0.89	3.928	204	14.957	1015.94
3	lluvia	8.289	5.944	0.83	14.104	269	15.826	1016.41
4	lluvia	8.756	6.978	0.83	11.045	259	15.826	1016.51

Descripción del modelo

Para poder desarrollar el modelo de regresión logística se utilizará el lenguaje de programación Python, debido a que es comúnmente utilizado por su simplicidad para desarrollar modelos de regresión.

Para facilitar el desarrollo del modelo se utilizará la herramienta de desarrollo web de Google Colab en combinación con Visual Studio Code. Se usaron los notebooks de Servidor Jupyter integrado al editor de Visual Studio Code, para ejecutar .

Proceso de desarrollo del modelo

Al principio luego de importar las librerías principales para nuestro proyecto, se detectaron las salidas introducidas en la columna ‘Tipo de Precipitación’.

```
dataset['Tipo de precipitacion'].value_counts()
```

```
1    17634
0     2366
Name: Tipo de precipitacion, dtype: int64
```

1 representa lluvia y 0 es la nieve. En esta celda dividimos los datos en columnas de entrada y salida. Se había reducido el número de filas del dataset.

```
Columnas_De_Entradas= ['Temperatura (C)', 'Temperatura aparente (C)',
       'Humedad', 'Velocidad viento (km/h)', 'Direccion del viento (grados)',
       'Visibilidad (km)', 'Presion (millibares)']
x=dataset[Columnas_De_Entradas]
y=dataset['Tipo de precipitacion'].astype('int')
```

Se preparan los datos para el entrenamiento y pruebas del método de Regresión Logística. Se entregarán el 80% de los datos para predecir los faltantes.

```
X_Entrenamiento, X_Prueba, y_Entrenamiento, y_Prueba= train_test_split(x,y, test_size=0.2, random_state=4)
```

```
RL = LogisticRegression(C=0.01, solver='liblinear').fit(X_Entrenamiento, y_Entrenamiento)

y_Predecido=RL.predict(X_Prueba)

#Porcentaje de ajuste
ajuste = RL.score(X_Prueba,y_Prueba)
print('\nEl porcentaje de ajuste es: ', ajuste*100, '%\n')
```

```
El porcentaje de ajuste es:  99.0 %
```

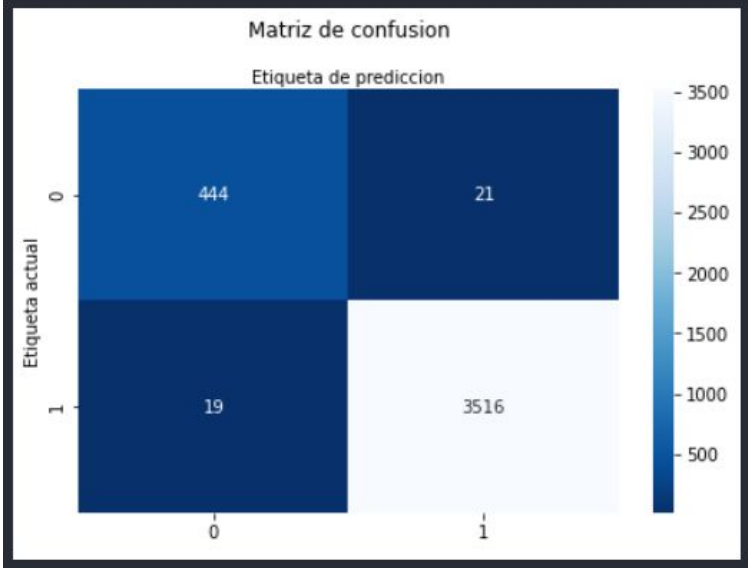
Se probó predecir con nuevos datos que ingresamos. El resultado es nieve:0. Matriz de confusión.Tabla de comparación con los datos predichos.

```
   X_usernp=pd.DataFrame({'Temperatura (C)': [-2.05], 'Temperatura aparente (C)': [-2.760],
          'Humedad': [0.80], 'Velocidad viento (km/h)': [2.750], 'Direccion del viento (grados)': [0.612813],
          'Visibilidad (km)':[0.305], 'Presion (millibares)': [0.901235]})

   y_Predecido=RL.predict(X_usernp)
   y_Predecido
```

```
array([0])
```

```
Matriz_De_Confusion= metrics.confusion_matrix(y_Prueba, y_Predecido)
Matriz_De_Confusion
Nombre_De_Clases=[0,1]
figura, datos=plt.subplots()
Marcas_De_Clases=np.arange(len(Nombre_De_Clases))
plt.xticks(Marcas_De_Clases, Nombre_De_Clases)
plt.yticks(Marcas_De_Clases, Nombre_De_Clases)

sea.heatmap(pd.DataFrame(Matriz_De_Confusion), annot=True, cmap='Blues_r', fmt='g')
datos.xaxis.set_label_position('top')
plt.tight_layout()
plt.title('Matriz de confusion', y=1.1)
plt.ylabel('Etiqueta actual')
plt.xlabel('Etiqueta de prediccion')
print("exactitud", metrics.accuracy_score(y_Prueba, y_Predecido))
```

```
exactitud 0.99
```



```
pd.DataFrame({'actual': y_Prueba,
              'prediccion':y_Predecido,
              'diferencia': y_Prueba-y_Predecido})
```

	actual	prediccion	diferencia
1919	0	0	0
6624	1	1	0
3858	1	1	0
5053	1	1	0
8745	1	1	0
...	...	...	...
3469	0	0	0
12015	1	1	0
10981	1	1	0
18582	1	1	0
10128	1	1	0

4000 rows × 3 columns

Métodos de regresión complementarios:Método de regresión polinómica

```
from sklearn.preprocessing import PolynomialFeatures

poly=PolynomialFeatures(degree=4)
X_poly=poly.fit_transform(X_Entrenamiento)

poly.fit(X_poly, y_Entrenamiento)
lin2= LinearRegression()
lin2.fit(X_poly, y_Entrenamiento)

#Error de predicción
predicciondos=lin2.predict(poly.fit_transform(X_Prueba))
#Calculando el error
np.mean((predicciondos- y_Prueba)**2)
```

```
0.02205414712221019
```

```
pd.DataFrame({'actual': y_Prueba,
              'predicción':predicciondos,
              'diff':(y_Prueba-predicciondos)})
```

	actual	predicción	diff
1919	0	-0.263512	0.263512
6624	1	0.895137	0.104863
3858	1	1.077672	-0.077672
5053	1	1.003855	-0.003855
8745	1	0.974713	0.025287
...	...	...	...
3469	0	0.173608	-0.173608
12015	1	1.085457	-0.085457
10981	1	0.969620	0.030380
18582	1	0.984018	0.015982
10128	1	1.004810	-0.004810

4000 rows × 3 columns

Método de regresión del árbol de decisión CART.

```
from  sklearn.tree import DecisionTreeRegressor
RGSOR=DecisionTreeRegressor(random_state=0)
RGSOR.fit(X_Entrenamiento, y_Entrenamiento)
```

```
DecisionTreeRegressor(random_state=0)
```

```
predicciontres=RGSOR.predict(X_Prueba)
np.mean((predicciontres-y_Prueba)**2)
```

```
0.0
```

```
pd.DataFrame({'actual': y_Prueba,
              'prediccion': predicciontres,
              'diff':(y_Prueba-predicciontres)})
```

	actual	prediccion	diff
1919	0	0.0	0.0
6624	1	1.0	0.0
3858	1	1.0	0.0
5053	1	1.0	0.0
8745	1	1.0	0.0
...	...	...	...
3469	0	0.0	0.0
12015	1	1.0	0.0
10981	1	1.0	0.0
18582	1	1.0	0.0
10128	1	1.0	0.0

4000 rows x 3 columns

Método de bosque aleatorio con departamento máximo=10

```
from sklearn.ensemble import RandomForestRegressor
regr=RandomForestRegressor(max_depth=10, random_state=0, n_estimators=100)
regr.fit(X_Entrenamiento, y_Entrenamiento)
```

```
RandomForestRegressor(max_depth=10, random_state=0)
```

```
prediccioncuatro=regr.predict(X_Prueba)
np.mean((prediccioncuatro-y_Prueba)**2)
```

```
0.0
```

```
pd.DataFrame({'actual': y_Prueba,
              'predicción': prediccioncuatro,
              'diff':(y_Prueba-prediccioncuatro)})
```

	actual	predicción	diff
1919	0	0.0	0.0
6624	1	1.0	0.0
3858	1	1.0	0.0
5053	1	1.0	0.0
8745	1	1.0	0.0
...	...	...	...
3469	0	0.0	0.0
12015	1	1.0	0.0
10981	1	1.0	0.0
18582	1	1.0	0.0
10128	1	1.0	0.0

4000 rows × 3 columns

Resultados obtenidos

Los **verdaderos positivos** (VP) son aquellas filas que fueron clasificados correctamente como 0 /nieve por el modelo.

Los **verdaderos negativos** (VN) corresponden a la cantidad de filas que fueron clasificados correctamente como 1/lluvia por el modelo.

Los **falsos negativos** (FN) es la cantidad de filas que fueron clasificados incorrectamente como 1/lluvia.

Los **falsos positivos** (FP) indican la cantidad de filas que fueron clasificados incorrectamente como 0/nieve.

El porcentaje de ajuste que se midió para el modelo es 99%. El error es del 1% con 40 datos no predecidos correctamente.

Para poder evitar el sobreajuste del modelo se redujeron el número de filas a 20,000. También se redujo el número de columnas, eliminando aquellas de contenido literal.

Conclusiones

El porcentaje de ajuste que se obtuvo en el modelo produjo resultados altos, lo que califica a la regresión logística como óptima. Se trata de un modelo eficiente que permite predecir a partir de nuevos datos con mejor exactitud y se puede utilizar