Taller 1 – Introducción a Data Science

Modelos de Regresión para la Predicción de Demanda de Bicicletas Compartidas

Descripción

Los sistemas de bicicletas compartidas representan una nueva generación de transporte urbano sostenible que ha revolucionado la movilidad en ciudades alrededor del mundo. Estos sistemas generan grandes volúmenes de datos que permiten entender patrones de movilidad urbana, comportamiento de usuarios y factores que influyen en la demanda de transporte alternativo. La capacidad de predecir con precisión la demanda de bicicletas tiene implicaciones directas en la planificación operativa, distribución de recursos y satisfacción del usuario.

En este taller, exploraremos cómo factores ambientales, temporales y sociales influyen en los patrones de uso de bicicletas compartidas. Aplicaremos técnicas de Análisis Exploratorio de Datos, Ingeniería de Características, y Modelos de Regresión para predecir la demanda horaria de bicicletas utilizando datos reales del sistema Capital Bikeshare de Washington D.C., que abarca dos años de operación (2011-2012).

Referencia del dataset: Fanaee-T, Hadi, and Gama, Joao, "Event labeling combining ensemble detectors and background knowledge", Progress in Artificial Intelligence (2013): pp. 1-15, Springer Berlin Heidelberg.

Objetivos

1. Diseñar características temporales significativas a partir de datos de fecha/hora para capturar patrones de demanda.
2. Aplicar técnicas de exploración de datos y visualización para entender la influencia de factores climáticos y temporales en la demanda.
3. Modelar el conjunto de datos utilizando técnicas de regresión lineal, incluyendo Regresión Lineal Múltiple, LASSO, y Ridge.
4. Optimizar el rendimiento de los modelos mediante validación cruzada y optimización de hiperparámetros.

Entregables

- Un cuaderno interactivo *Jupyter Notebook* con todo el código fuente, resultados, y análisis.
- El cuaderno debe utilizar apropiadamente celdas de *markdown* y de código.
- Todas las visualizaciones deben incluir títulos, etiquetas de ejes y leyendas cuando sea apropiado.

Fecha de Entrega

28 de Septiembre de 2025 a las 23:59.

Conjunto de Datos

Se proporcionan un único archivo CSV en Campus Virtual:

- **bike_sharing_data.csv:** Datos de entrenamiento de demanda por hora de todos los días del mes.

Variables disponibles:

- **datetime**: Fecha y hora del registro
- **season**: Estación (1=primavera, 2=verano, 3=otoño, 4=invierno)
- **holiday**: Si el día es feriado (0=no, 1=sí)
- **workingday**: Si es día laboral (0=no, 1=sí)
- **weather**: Condición climática (1=despejado, 2=nublado, 3=lluvia ligera, 4=lluvia fuerte)
- **temp**: Temperatura en Celsius

- **atemp**: Sensación térmica en Celsius
- **humidity**: Humedad relativa (0-100)
- **windspeed**: Velocidad del viento
- **casual**: Usuarios no registrados
- **registered**: Usuarios registrados
- **count**: Total de alquileres- **Variable objetivo**

Nota importante: Para este taller, trabajaremos prediciendo el logaritmo del conteo total (log(count + 1)) para manejar la naturaleza sesgada de la distribución de demanda. Por supuesto, verificaremos que esto entregue buenos resultados.

Requerimientos Mínimos

1. Ingeniería de Características y Exploración [25 puntos]

a) División de los datos en train/validation/test [5 puntos]:

- Dividir el conjunto de datos de la siguiente forma.
- El conjunto de entrenamiento irá del día 1 al día 14 (inclusive), lo que cubre dos semanas del mes.
- El conjunto de validación (para optimización de hiperparámetros) irá del día 15 al día 21 (inclusive), lo que permite hacer prueba durante otra semana.
- El conjunto de pruebas final irá del día 22 al fin de mes y solo se utilizará para la validación final.
- **Importante**: Cualquier transformación debe ser ajustada solo con datos de **entrenamiento** y luego aplicada a los datos de validación/prueba.

b) Extracción de características temporales [5 puntos]:

Desde la columna datetime, extraer las siguientes características:

- Hora del día
- Día de la semana (0=Lunes, 6=Domingo)
- Mes
- Año

Adicionalmente, crear las siguientes variables derivadas:

- is_weekend: Variable binaria (1 si es sábado o domingo), notar que no es necesariamente lo mismo que workingday por los feriados, aunque quizás no aporta mucho.
- hour_category: Categorizar las horas en períodos del día (puede considerar otros puntos de quiebre):
    - Madrugada (0-5)
    - Mañana (6-11)
    - Tarde (12-17)
    - Noche (18-23)
- is_rush_hour: Variable binaria para horas de mayor demanda (7-9 AM y 5-7 PM en días laborales)

c) Análisis exploratorio [10 puntos]:

- Generar estadísticas descriptivas para todas las variables.
- Crear una matriz de correlación y visualizarla con un heatmap.
- Graficar la distribución de count y de log(count + 1) para justificar la transformación. En principio debería haber un mejor comportamiento con el enfoque logarítmico, pero se debe revisar que efectivamente así.
- Crear al menos 4 visualizaciones que muestren:
    - Demanda promedio por hora del día
    - Demanda promedio por día de la semana
    - Relación entre temperatura y demanda

- Boxplots de demanda por condición climática

d) Preparación de datos [5 puntos]:

- Identificar y documentar valores faltantes o anómalos.
- Aplicar la transformación logarítmica a la variable objetivo. ***Se recomienda considerar ambos modelos (modelo base con count y con la versión transformada).***
- Normalizar las variables numéricas apropiadamente.
- Se recuerda nuevamente que cualquier transformación debe ser ajustada solo con datos de entrenamiento y luego aplicada a los datos de validación/prueba con los mismos valores usados en entrenamiento.

2. Regresión Lineal Base [20 puntos]

a) Modelo inicial [10 puntos]:

Entrenar un modelo de regresión lineal usando SOLO las variables originales (sin las características temporales creadas):

- Variables: season, holiday, workingday, weather, temp, atemp, humidity, windspeed
- Evaluar con MSE, RMSE, MAE y R^2 sobre el conjunto de validación.
- Interpretar los coeficientes más importantes

b) Modelo mejorado [10 puntos]:

Entrenar un segundo modelo incluyendo TODAS las características creadas:

- Comparar métricas con el modelo base sobre el conjunto de validación.
- Crear un gráfico de barras mostrando la importancia de los coeficientes
- Analizar multicolinealidad entre temp y atemp usando VIF (Factor de Inflación de la Varianza)
- Graficar predicciones vs valores reales para ambos modelos

3. Regularización: LASSO y Ridge [20 puntos]

a) Implementación de Ridge [10 puntos]:

- Implementar regresión Ridge con un rango de valores alpha entre 10^-3 y 10^3
- Para optimizar hiperparámetros, usar el conjunto de validación. También se puede usar validación cruzada especializada con TimeSeriesSplit (5 splits) con todos los datos de training y validation combinados. Puede elegir entre ambos métodos, se recomienda la opción con TimeSeriesSplit dado que es lo que se recomendaría en la práctica, pero la otra opción es más simple.
- Graficar cómo cambian los coeficientes con diferentes valores de alpha
- Reportar el alpha óptimo y las métricas de evaluación sobre el conjunto de validación o el promedio de validación cruzada.

b) Implementación de LASSO [10 puntos]:

- Implementar regresión LASSO con el mismo rango de valores alpha
- Usar la misma estrategia de validación que para Ridge
- Identificar qué variables son eliminadas por LASSO (coeficientes = 0)
- Comparar el número de características seleccionadas vs Ridge
- Crear una tabla comparativa de métricas entre Lineal, Ridge y LASSO

4. Elastic Net y Análisis Avanzado [20 puntos]

a) Elastic Net [10 puntos]:

- Implementar Elastic Net optimizando tanto alpha como l1_ratio
- Usar l1_ratio en [0.1, 0.3, 0.5, 0.7, 0.9]
- Documentar los hiperparámetros óptimos encontrados

- Comparar con todos los modelos anteriores sobre el conjunto de validación o el promedio de validación cruzada.

b) Análisis de residuos y patrones temporales [10 puntos]:

- Graficar residuos vs predicciones para detectar heterocedasticidad
- Analizar si hay patrones en los residuos por hora del día o día de la semana
- Crear un gráfico de serie temporal mostrando predicciones vs valores reales para una semana completa
- Realizar test de normalidad de residuos (Shapiro-Wilk o Q-Q plot)

5. Conclusiones y Recomendaciones [15 puntos]

- Crear una tabla resumen con todos los modelos y sus métricas principales
- Identificar los 5 factores más importantes para predecir la demanda
- Discutir limitaciones del análisis:
    - Problema de extrapolación temporal
    - Variables no consideradas (eventos especiales, cambios en infraestructura)
    - Suposiciones de los modelos lineales
- Proponer al menos 3 mejoras concretas para futuros análisis
- Recomendaciones operativas para el sistema de bicicletas basadas en los hallazgos

Créditos Extra

6. Modelado por tipo de usuario [5 puntos]:

- Crear modelos separados para predecir log(casual + 1) y log(registered + 1)
- Comparar qué factores influyen más en cada tipo de usuario
- Analizar si la suma de las predicciones separadas mejora la predicción total

7. Interacciones y no linealidades [5 puntos]:

- Crear términos de interacción:
    - temperatura × humedad
    - hora × día laboral
    - clima × estación
- Implementar características polinómicas para temperatura (cuadráticas y cúbicas)
- Evaluar si estas características mejoran el modelo
- Visualizar las relaciones no lineales encontradas

Consideraciones de Evaluación

- El taller se puede realizar en grupos de 1 o 2 estudiantes.
- Total: 100 puntos base + 10 puntos de crédito extra.
- Los créditos extra solo se evalúan si los puntos base están completos.
- Escala de evaluación al 60%.
- **Importante**: El notebook debe ejecutarse completamente sin errores. Se evaluará hasta el primer error encontrado.
- Los análisis deben ser claros y bien argumentados.
- Todas las decisiones de modelado deben estar justificadas.
- El formato de entrega será el siguiente "NombreApellido1_NombreApellido2.ipynb" (sin tildes) mediante plataforma de campus virtual.
    - Ejemplo: PedroRojas_JosePerez.ipynb
- **Política de días de atraso:** Durante el semestre existen días de atraso para las entregas de talleres según lo indicado en la primera clase del curso (ver Diapositivas 01). Considerar los siguientes puntos:

- Cada día se consumirá automáticamente al utilizar 1 minuto del día siguiente a la fecha de entrega predefinida, después de consumirse todos los días de atraso, se dará paso al descuento de nota definido a continuación.
- A partir de las 00:00 hrs del día siguiente del día de entrega comenzará un descuento de 10 décimas por cada hora de retraso sobre su nota.
- Ejemplo: Si el límite de entrega es el 29 de agosto a las 23:59 hrs y la tarea es entregada el 30 de agosto a las 01:00 hrs, la nota máxima posible es 6.0.

Notas Importantes

Sobre la transformación logarítmica: Usar np.log1p() para transformar (equivale a log(x+1)). Usar np.expm1() para revertir la transformación (equivale a exp(x)-1). Esto evita problemas con valores cero.

Sobre la evaluación de modelos: Reportar métricas tanto en escala logarítmica como en escala original (*count*). Para la escala original, primero revertir la transformación y luego calcular las métricas.

Sobre validación temporal: TimeSeriesSplit respeta el orden temporal de los datos. No usar división aleatoria para datos temporales.

Sobre normalización: El StandardScaler (o cualquier transformación) debe ser ajustado **SOLO** con datos de entrenamiento. Aplicar la transformación aprendida a los datos de prueba (sin reajustar).

Si utiliza IA generativa, documente todos los prompts a lo largo del notebook como documentación.

Ejemplo de Uso de Test de Shapiro-Wilk y Q-Q Plots para verificar normalidad.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# Generar datos de ejemplo
np.random.seed(42)
datos_normal = np.random.normal(100, 15, 100)  # Distribución normal
datos_no_normal = np.random.exponential(10, 100)  # Distribución exponencial

# Test de Shapiro-Wilk (una línea)
stat, p_value = stats.shapiro(datos_normal)
print(f"Shapiro-Wilk: estadístico={stat:.4f}, p-valor={p_value:.4f}")
print(f"¿Datos normales? {'Sí' if p_value > 0.05 else 'No'} (α=0.05)")

# Q-Q Plot (prácticamente una línea)
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 4))

# Q-Q plot para datos normales
stats.probplot(datos_normal, dist="norm", plot=ax1)
ax1.set_title("Q-Q Plot - Datos Normales")

# Q-Q plot para datos no normales
stats.probplot(datos_no_normal, dist="norm", plot=ax2)
ax2.set_title("Q-Q Plot - Datos Exponenciales")

plt.tight_layout()
plt.show()
```