Aprendizaje Automatizado

Practica 2: Modelo de Regresion (Renta de bicicletas (SEUL)

Nombre Cynthia Selene Martínez Espinoza **Matricula** 1011238

Carga de Librerias / funciones

In [1]:
```
pip install numpy pandas matplotlib scikit-learn
```

```
Requirement already satisfied: numpy in c:\users\pc\anaconda3\lib\site-packages (1.2
6.4)
Requirement already satisfied: pandas in c:\users\pc\anaconda3\lib\site-packages (2.
1.4)
Requirement already satisfied: matplotlib in c:\users\pc\anaconda3\lib\site-packages
(3.8.0)
Requirement already satisfied: scikit-learn in c:\users\pc\anaconda3\lib\site-packag
es (1.2.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\pc\anaconda3\lib\s
ite-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\pc\anaconda3\lib\site-packag
es (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\pc\anaconda3\lib\site-pack
ages (from pandas) (2023.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\pc\anaconda3\lib\site-pa
ckages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\pc\anaconda3\lib\site-packag
es (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\pc\anaconda3\lib\site-p
ackages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\pc\anaconda3\lib\site-p
ackages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\pc\anaconda3\lib\site-pac
kages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\pc\anaconda3\lib\site-packa
ges (from matplotlib) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\pc\anaconda3\lib\site-pa
ckages (from matplotlib) (3.0.9)
Requirement already satisfied: scipy>=1.3.2 in c:\users\pc\anaconda3\lib\site-packag
es (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in c:\users\pc\anaconda3\lib\site-packa
ges (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\pc\anaconda3\lib\sit
e-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\users\pc\anaconda3\lib\site-packages
(from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [32]:
```python
#Importar librerias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier
from sklearn.metrics import mean_squared_error, accuracy_score, r2_score
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
```

Carga de Datos

In [2]:
```
# Leer los datos de archivo csv, typed_uanl.csv con el URL
url =  "C:/Users/PC/Documents/GitHub/GitFlow-en-Github/ML003/Practica 2/Practica/Re
df = pd.read_csv(url)
```

Explorando de Datos

In [3]:
```
# Resumen estadístico de los datos
summary_stats = df.describe()
# Verificación de valores nulos
missing_values = df.isnull().sum()
```

In [5]:
```
summary_stats, missing_values
```

Out[5]:
```
(       Rented Bike Count         Hour  Temperature(°C)  Humidity(%)  \
count        8760.000000  8760.000000      8760.000000  8760.000000
mean          704.602055    11.500000        12.882922    58.226256
std           644.997468     6.922582        11.944825    20.362413
min             0.000000     0.000000       -17.800000     0.000000
25%           191.000000     5.750000         3.500000    42.000000
50%           504.500000    11.500000        13.700000    57.000000
75%          1065.250000    17.250000        22.500000    74.000000
max          3556.000000    23.000000        39.400000    98.000000

       Wind speed (m/s)  Visibility (10m)  Dew point temperature(°C)  \
count       8760.000000       8760.000000                8760.000000
mean           1.724909       1436.825799                   4.073813
std            1.036300        608.298712                  13.060369
min            0.000000         27.000000                 -30.600000
25%            0.900000        940.000000                  -4.700000
50%            1.500000       1698.000000                   5.100000
75%            2.300000       2000.000000                  14.800000
max            7.400000       2000.000000                  27.200000

       Solar Radiation (MJ/m2)  Rainfall(mm)  Snowfall (cm)
count              8760.000000   8760.000000    8760.000000
mean                  0.569111      0.148687       0.075068
std                   0.868746      1.128193       0.436746
min                   0.000000      0.000000       0.000000
25%                   0.000000      0.000000       0.000000
50%                   0.010000      0.000000       0.000000
75%                   0.930000      0.000000       0.000000
max                   3.520000     35.000000       8.800000  ,
Date                         0
Rented Bike Count            0
Hour                         0
Temperature(°C)              0
Humidity(%)                  0
Wind speed (m/s)             0
Visibility (10m)             0
Dew point temperature(°C)    0
Solar Radiation (MJ/m2)      0
Rainfall(mm)                 0
Snowfall (cm)                0
Seasons                      0
Holiday                      0
Functioning Day              0
dtype: int64)
```

In [4]:
```
df.head(5)
```

Out[4]:

	Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew p temperature
0	01/12/2017	254	0	-5.2	37	2.2	2000	-
1	01/12/2017	204	1	-5.5	38	0.8	2000	-
2	01/12/2017	173	2	-6.0	39	1.0	2000	-
3	01/12/2017	107	3	-6.2	40	0.9	2000	-
4	01/12/2017	78	4	-6.0	36	2.3	2000	-

Encontrar al mejor modelo, con Validacion Cruzada

In [5]:
```
# Preprocesar los datos: convertir columnas categóricas a numéricas y manejar valor
df['Seasons'] = df['Seasons'].astype('category').cat.codes
df['Holiday'] = df['Holiday'].astype('category').cat.codes
df['Functioning Day'] = df['Functioning Day'].astype('category').cat.codes
```

In [6]:
```
# Selección de características y variable objetivo
features = [
    'Hour', 'Temperature(°C)', 'Humidity(%)', 'Wind speed (m/s)',
    'Visibility (10m)', 'Dew point temperature(°C)',
    'Solar Radiation (MJ/m2)', 'Rainfall(mm)', 'Snowfall (cm)',
    'Seasons', 'Holiday', 'Functioning Day'
]
X = df[features]
y = df['Rented Bike Count']
```

In [9]:
```
modelos = {
    'Regresion Lineal': LinearRegression(),
    'Regresion Polynomica (grado=2)': make_pipeline(PolynomialFeatures(degree=2), S
    'Regresion cresta': make_pipeline(StandardScaler(), Ridge()),
    'Regresion KNN': make_pipeline(StandardScaler(), KNeighborsRegressor(n_neighbor
    'Regresion Arbol de desicion': DecisionTreeRegressor()
}
```

In [14]:
```
# Validación cruzada
resultados = {}
for nombre, modelo in modelos.items():
    r2 = cross_val_score(modelo, X, y, cv=5, scoring='r2').mean()
    resultados[nombre] = r2

# Encontrar el mejor modelo
```

```
mejor_modelo_nombre = max(resultados, key=resultados.get)
mejor_modelo = modelos[mejor_modelo_nombre]
```

In [16]:
```
# Mostrar resultados
print(f"Resultados de la Validación Cruzada (Accuracy): {resultados}")
print(f"El mejor modelo es: {Mejor_Modelo_Nombre} con una Accuracy de {resultados[M
```

```
Resultados de la Validación Cruzada:
Regresion Lineal: R^2 = -0.4594
Regresion Polynomica (grado=2): R^2 = -2588720634079966480302080.0000
Regresion cresta: R^2 = -0.4590
Regresion KNN: R^2 = 0.2455
Regresion Arbol de desicion: R^2 = 0.3324
```

In [18]:
```
print(f"\nEl mejor modelo es: {mejor_modelo_nombre} con R^2 = {resultados[mejor_mod
```

```
El mejor modelo es: Regresion Arbol de desicion con R^2 = 0.3324
```

Entrenar mejor modelo encontrado en validacion cruzada

In [19]:
```
# Entrenar el mejor modelo en todo el conjunto de datos
mejor_modelo.fit(X, y)
```

Out[19]:
```
DecisionTreeRegressor
DecisionTreeRegressor()
```

In [22]:
```
# Dividir en conjunto de entrenamiento y prueba
X_entrenamiento, X_prueba, y_entrenamiento, y_prueba = train_test_split(X, y, test_

# utilizamos el mejor modelo de regresión,  árbol de decisión
modelo = DecisionTreeRegressor()

# Validación cruzada
r2_datos = cross_val_score(modelo, X_entrenamiento, y_entrenamiento, cv=5, scoring=

# Entrenar el modelo en todo el conjunto de datos de entrenamiento
modelo.fit(X_entrenamiento, y_entrenamiento)
```

Out[22]:
```
DecisionTreeRegressor
DecisionTreeRegressor()
```

In [27]:
```
# Evaluar el modelo en el conjunto de prueba
y_proyeccion = modelo.predict(X_prueba)
prueba_r2 = r2_score(y_prueba, y_proyeccion)

# Mostrar resultados
print(f"Resultados de la Validación Cruzada (R^2): {r2_datos}")
print(f"R^2 promedio: {r2_datos.mean():.4f}")
print(f"R^2 en el conjunto de prueba: {prueba_r2:.4f}")
```

```
Resultados de la Validación Cruzada (R^2): [0.73164226 0.73095454 0.7676254  0.76933
193 0.74267066]
R^2 promedio: 0.7484
R^2 en el conjunto de prueba: 0.6985
```

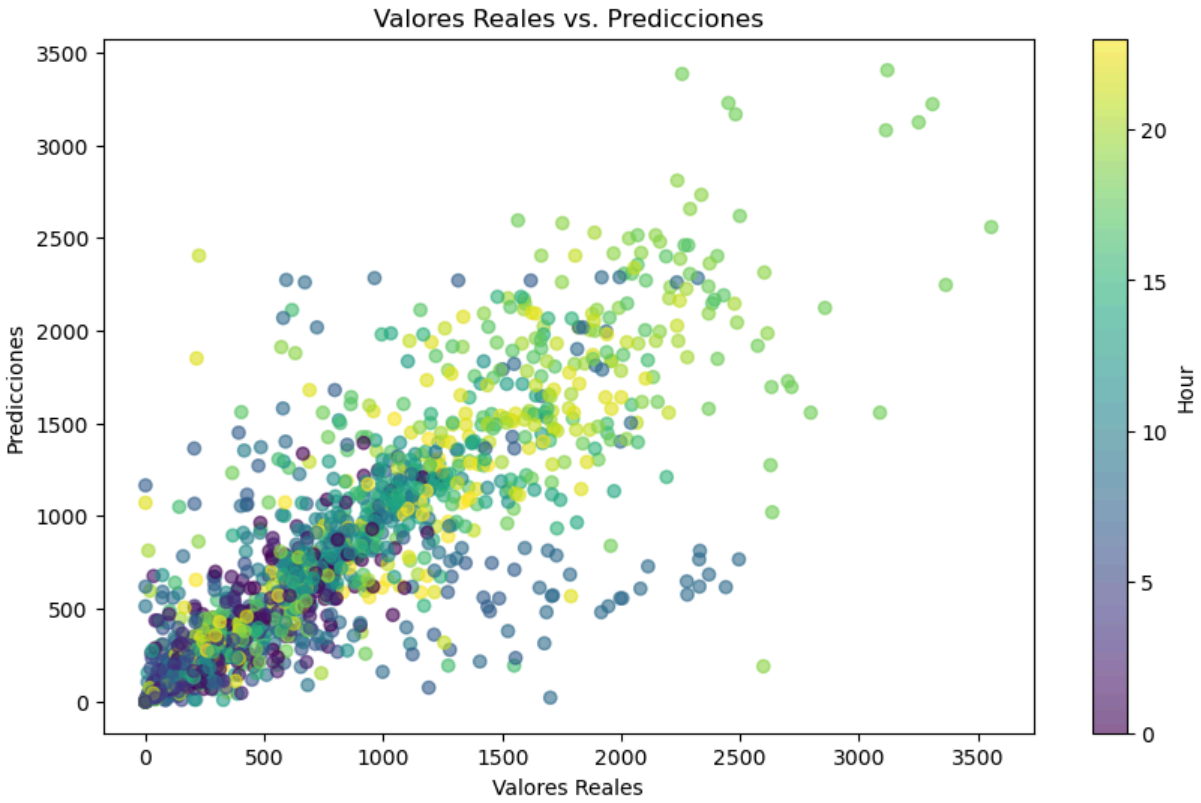
In [28]:
```python
#Evaluar el Modelo
#Evaluamos el rendimiento del modelo calculando el error cuadrático medio (MSE) y e
# Calcular el error cuadrático medio
mse = mean_squared_error(y_prueba, y_proyeccion)
print(f"Mean Squared Error: {mse}")

# Calcular el coeficiente de determinación (R²)
#r2 = r2_score(y_test, y_pred)
print(f"R² Score: {prueba_r2}")
```

```
Mean Squared Error: 125606.54623287672
R² Score: 0.6985295080936884
```

Grafica

In [35]:
```python
# Crear el gráfico de dispersión coloreando según la variable 'Hour'
plt.figure(figsize=(10, 6))
scatter = plt.scatter(y_prueba, y_proyeccion, c=X_prueba['Hour'], cmap='viridis', a
plt.xlabel('Valores Reales')
plt.ylabel('Predicciones')
plt.title('Valores Reales vs. Predicciones')
# Añadir barra de color
cbar = plt.colorbar(scatter)
cbar.set_label('Hour')
# Guardar y mostrar el gráfico
plt.savefig("C:/Users/PC/Documents/GitHub/GitFlow-en-Github/ML003/Practica 2/Practi
plt.show()
plt.close()
```



Analisis de resultados

Error Cuadrático Medio (MSE): Un MSE de 125606.54 indica que, en promedio, las predicciones del modelo tienen un error cuadrático bastante alto. Esto sugiere que las predicciones no son muy precisas.

Coeficiente de Determinación (R^2): Un R^2 de 0.698 Este valor indica que el modelo de árbol de decisión explica aproximadamente el 69.8% de la variabilidad en la renta de bicicletas. Esto sugiere que el modelo tiene un rendimiento razonablemente bueno, pero hay espacio para mejoras.

La hora del día, la temperatura, la humedad y otros factores meteorológicos (como la lluvia y la nieve) son determinantes importantes en la renta de bicicletas. El modelo identifica estos factores como los más influyentes. Impacto de las Condiciones Meteorológicas: Las condiciones meteorológicas adversas, como altas temperaturas, lluvia y nieve, tienden a reducir la demanda de bicicletas.

In [ ]: