Resumen del trabajo

Nombre: Lopez Chavez Pablo

CU: 111-386

CARRERA : Ing. En Ciencias de la computación

Primero importamos las librerías y damos una semilla para que se genere las estaturas randon, en un array se guarda estaturas con valor entre el mínimo ya sea 1.5 y el máximo 2.1 y serán 100 estaturas, y se crea un array peso

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeature
from sklearn.metrics import mean_squared_error
np.random.seed(42)


estaturas = np.random.uniform(1.5, 2.1, 100)


pesos = []
```

luego mediante un for que va recorriendo las estaturas con todos los valores de estatura, calcula el peso mínimo y el máximo y luego con ese peso saca un numero randon entre esos dos pesos y se agrega al array peso (hay otra manera con mayor fidelidad que usa una distribución normal para simular el peso, con una desviación estándar del 10% del rango)

Forma mala

```
for estatura in estaturas:
    # Calcular el peso mínimo y máximo según el IMC (Índice de Masa Corporal normal
    peso_min = 18.5 * (estatura ** 2)
    peso_max = 24.9 * (estatura ** 2)

    peso = np.random.uniform(peso_min, peso_max)
    pesos.append(peso)
```

Forma buena

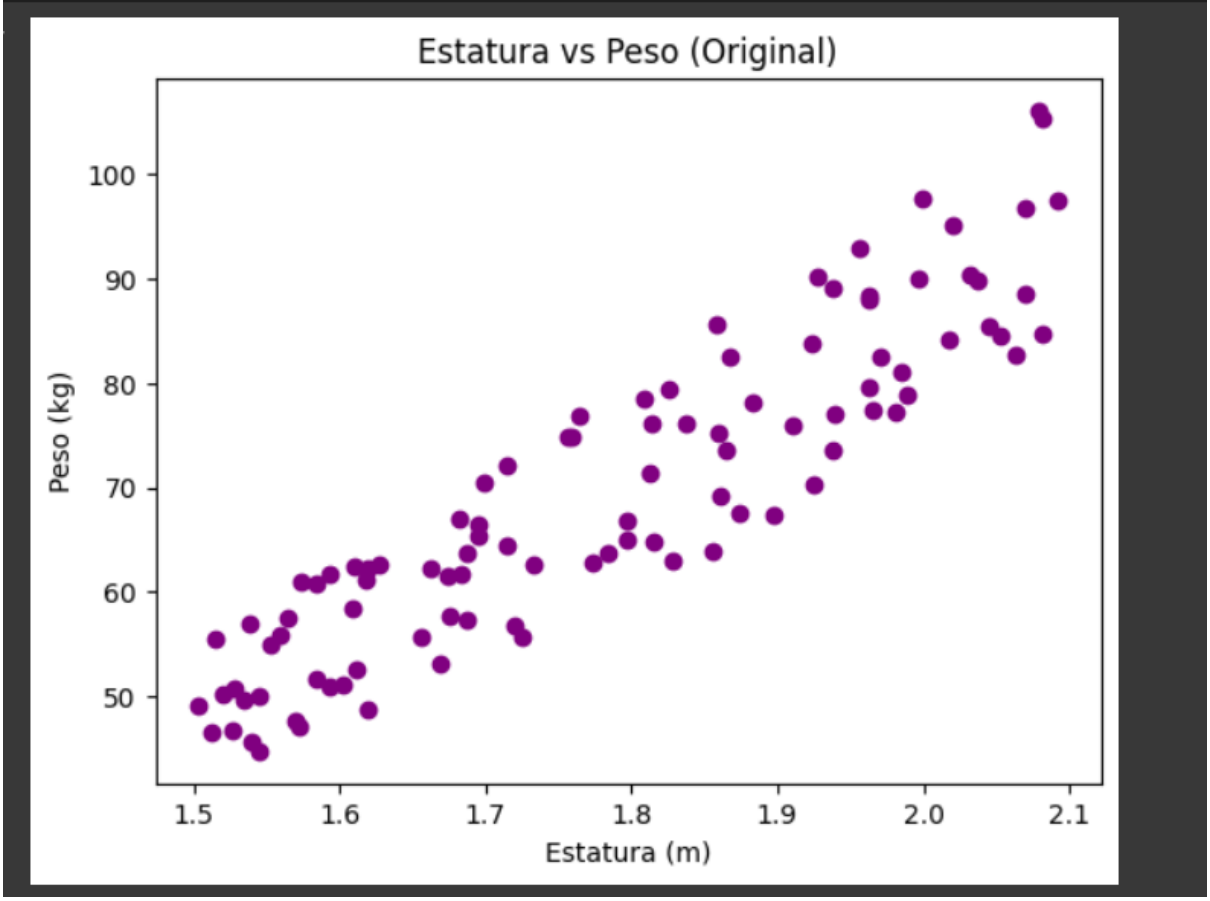
```
    # Calcular peso mínimo y máximo de mejor manera asi para que el modelo sea mas preciso
for estatura in estaturas:
    peso_min = 18.5 * (estatura ** 2)
    peso_max = 24.9 * (estatura ** 2)

    #Usar una distribución normal para simular el peso, con una desviación estándar del 10% del rango
    desviacion = (peso_max - peso_min) * 0.1
    peso = np.random.normal(loc=(peso_min + peso_max) / 2, scale=desviacion)

    #Asegurarse de que el peso esté dentro del rango
    peso = np.clip(peso, peso_min, peso_max)

    # Agregar el peso a la lista
    pesos.append(peso)
```

, luego se crea el dataframe con las estaturas y pesos generados con las Columnas Estatura (m) y Peso (kg) y lo mostramos

```
data = pd.DataFrame({
    'Estatura (m)': estaturas,
    'Peso (kg)': pesos
})
#imprimimos los pesos y estaturas
print(data)
```

Para imprimir sacamos los valores del dataframe de las respectivas columnas Estatura y Peso para poner en la grafica como puntos de color morado, se le pone nombre a la gráfica, y se da los parámetros para el eje “X” que en este caso es la estatura “Y” el eje y que es el Peso y mostramos el grafico

```
plt.scatter(data['Estatura (m)'], data['Peso (kg)'], color='purple')
plt.title('Estatura vs Peso (Original)')  # Título del gráfico
plt.xlabel('Estatura (m)')  # Etiqueta del eje X
plt.ylabel('Peso (kg)')  # Etiqueta del eje Y
plt.show()  # Mostrar el gráfico
```



Luego ajustamos los datos para la curva polinómica de grado 2 que en este caso tomaremos como Altura como “característica (x)” y Peso como “variable objetivo Y” por ende necesitamos adaptar a x para su procesamiento como que ya no sea un array con las estaturas si no una matriz con una columna y la variable “Y” no se toca

```
X = data['Estatura (m)'].values.reshape(-1, 1)
y = data['Peso (kg)'].values
print(X[:10])
```

```
[[1.72472407]
 [2.07042858]
 [1.93919637]
 [1.85919509]
 [1.59361118]
 [1.59359671]
 [1.53485017]
 [2.01970569]
 [1.86066901]
 [1.92484355]]
```

Una vez tenemos esto así transformamos la “X” para su procesamiento, pero primero creamos la polinómica de grado 2 tenemos que especificar el grado para que haga las transformaciones respectivas para “X” como ser agregar una columna de sesgo o intercepto, luego el valor original luego el valor cuadrático del valor original

```
[12] poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)
print(X_poly[:10])
```

```
[[1.         1.72472407 2.97467312]
 [1.         2.07042858 4.28667452]
 [1.         1.93919637 3.76048254]
 [1.         1.85919509 3.45660638]
 [1.         1.59361118 2.53959661]
 [1.         1.59359671 2.53955048]
 [1.         1.53485017 2.35576504]
 [1.         2.01970569 4.07921106]
 [1.         1.86066901 3.46208915]
 [1.         1.92484355 3.70502268]]
```

Se crea el modelo para la regresión lineal en este caso una parábola, y luego se le entrena con los valores de sesgo o intercepto con la pendiente que es la variable objetivo y

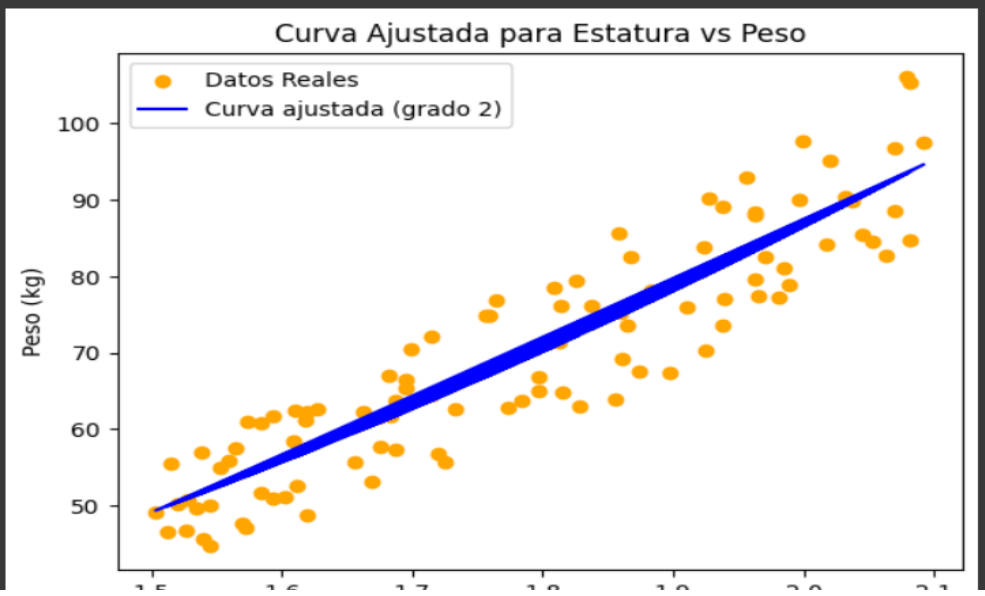
```
[13] model = LinearRegression()
     model.fit(X_poly, y)
```

Una vez que se tiene el modelo se intenta hacer la predicción con los valores de X_poly, luego mide el error cuadrático comparado las diferencias entre “y” y “y_pred”

```
[14] y_pred = model.predict(X_poly)

     mse = mean_squared_error(y, y_pred)
     print(f"Error cuadrático medio (MSE): {mse:.2f}")
```

Por ultimo mostramos de nuevo los puntos originales con un label para su mejor comprensión en la gráfica, luego creamos la curva o parábola con los valores de “X” y el las predicciones y de igual manera se le da un nombre

```
plt.scatter(data['Estatura (m)'], data['Peso (kg)'], color='orange', label='Datos F
plt.plot(data['Estatura (m)'], y_pred, color='blue', label='Curva ajustada (grado 2
plt.title('Curva Ajustada para Estatura vs Peso')
plt.xlabel('Estatura (m)')
plt.ylabel('Peso (kg)')
plt.legend()
plt.show()
```



Conclusiones:

En este caso el modelo tiene muchos errores por los datos generados de manera aleatoria, si se aplica la otra forma de generar pesos se mejora.

Este modelo se podría decir que es aprendizaje supervisado

Mejora de datos