Librerias

In [ ]:

```
import numpy as np
from pandas import read_csv
import matplotlib.pyplot as plt

from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

Importar Database

In [ ]:

```
df = read_csv('Advertisement.csv')
df.head()
```

Out[ ]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

Visualización de Datos

In [ ]:

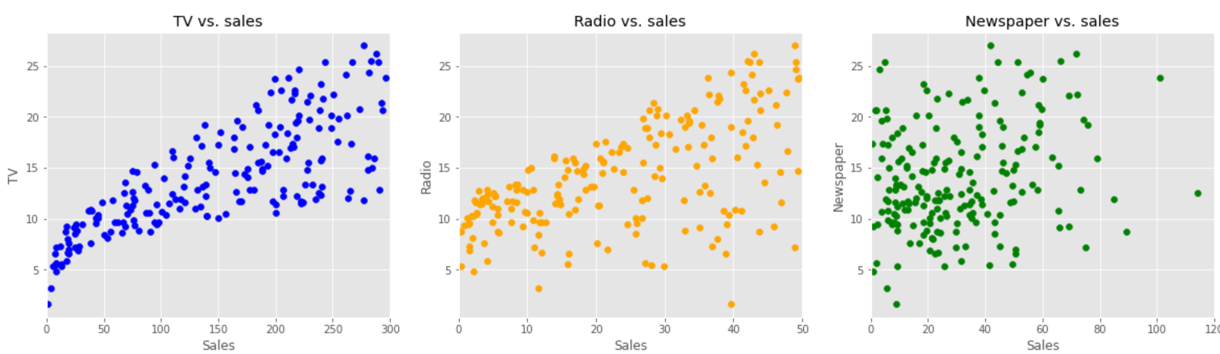
```
plt.style.use('ggplot')
fig, (ax1, ax2, ax3) = plt.subplots(1,3, figsize=(20, 5))

ax1.scatter(df['TV'], df['Sales'], c ="blue")
ax1.set_xlim(0,300)
ax1.set_title('TV vs. sales')
ax1.set_xlabel('Sales')
ax1.set_ylabel('TV')

ax2.scatter(df['Radio'], df['Sales'], c ="orange")
ax2.set_xlim(0,50)
ax2.set_title('Radio vs. sales')
ax2.set_xlabel('Sales')
ax2.set_ylabel('Radio')

ax3.scatter(df['Newspaper'], df['Sales'], c ="green")
ax3.set_xlim(0,120)
ax3.set_title('Newspaper vs. sales')
ax3.set_xlabel('Sales')
ax3.set_ylabel('Newspaper')

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



Variables Independientes y Dependientes

In [ ]:

```
#separar los otros atributos del atributo de predicción.
x = df.drop(["Sales", "Radio", "Newspaper"], axis=1)

#separe el atributo de predicción en Y para el entrenamiento del modelo.
y = df["Sales"]
```

Dividir Datos

In [ ]:

```
# splitting the data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, sh
```

Aplicar Modelo

In [ ]:

```
# creando un objeto de la clase LinearRegression
model = LinearRegression()

# Ajustando los datos de entrenamiento.
model.fit(x_train,y_train)

# Realizar predicion.
y_pred =  model.predict(x_test)
y_pred
```

Out[ ]:

```
array([ 8.80059198, 11.50793912, 13.09467996, 17.05446942, 17.80781819,
        9.70931893,  7.79769644,  7.91540718, 11.03709614, 18.29749489,
       11.10772258,  8.48512719, 17.54885455, 17.63360628, 12.79804888,
       18.45287307, 10.59921217, 13.17472327, 20.21853424, 19.30509886,
        8.7393824 ,  7.34568717, 12.66150442, 12.17182772, 15.59956461,
       12.0588254 , 14.13524294, 16.30582908,  8.18378768, 16.30582908,
        7.36922932, 10.22253778, 13.73031798, 16.09865817, 10.7781325 ,
       12.46375037,  8.34858272,  9.64810935, 18.29278646, 16.36703867])
```

Visualización Resultados

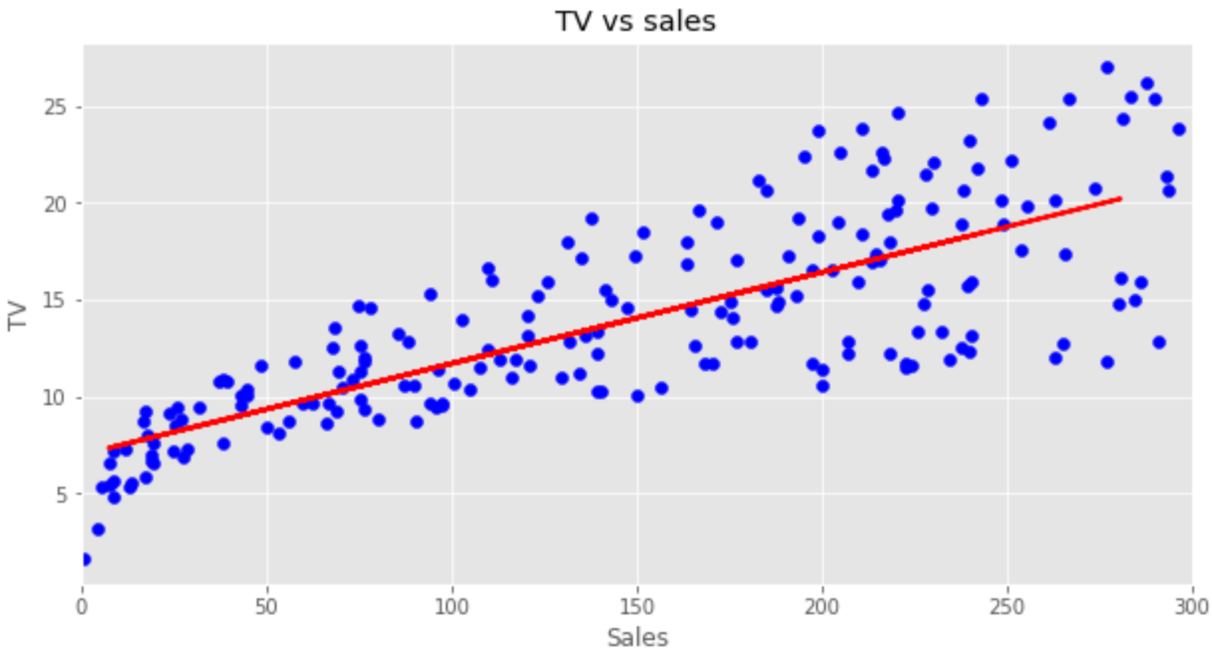
In [ ]:

```
#plt.style.use('seaborn')

fig = plt.subplots(figsize=(10,5))

plt.scatter(df['TV'], df['Sales'], c ="blue")
plt.plot(x_test, y_pred, color="red", linewidth=2)
plt.xlim(0,300)
plt.title('TV vs sales')
plt.xlabel('Sales')
plt.ylabel('TV')

plt.show()
```



Evaluacion del Modelo

In [ ]:

```
# Funcion para metrica de MAPE.
def mape_func(real, pred):
    real, pred = np.array(y_test), np.array(pred)
    return np.mean(np.abs((y_test - pred) / real)) * 100

# Evaluar valores reales versus valores estimados.
print("R2:", "{0:.4f}".format(r2_score(y_test, y_pred)),)
print("Adjust R2:", "{0:.4f}".format(1- (1-model.score(x, y))*(len(y)-1)/(len
print("MAPE:", "{0:.2f}".format(mape_func(y_test, y_pred)),"%")
```

```
R2: 0.6050
Adjust R2: 0.6095
MAPE: 18.75 %
```