1. **Dataset**
   Data Housing Boston diambil dari sumber Kaggle
   https://www.kaggle.com/datasets/altavish/boston-housing-dataset
   Data ini berisi tentang informasi yang dikumpulkan oleh Layanan Sensus AS mengenai berbagai faktor yang mempengaruhi harga rumah di berbagai lingkungan di kota Boston, Massachusetts, Amerika Serikat.

1. **Link GCollab**
   https://colab.research.google.com/drive/1kE2kJn1P8lcwHRaWlSfvYzIQmH1Lz4Yu?usp=sharing

2. **Membaca Data**

```
[ ]  import pandas as pd
```

Pertama, import terlebih dahulu library yang dibutuhkan untuk mengimport, menampilkan data, analisis data dan manipulasi data dalam bentuk DataFrame, yakni bernama "pandas" dan memberinya alias "pd."

```
[ ]  # membaca data

     data = pd.read_csv('HousingData.csv')
     df = pd.DataFrame(data)
     print(df)
```

Gunakan fungsi **read_csv** dari Pandas untuk membaca data dari file CSV. Disini, file saya beri nama "HousingData.csv." Fungsi ini membaca data dari file tersebut dan menyimpannya dalam bentuk objek Pandas DataFrame dengan nama "data".

Setelah data dibaca, **df = pd.DataFrame(data)** mengonversinya menjadi DataFrame Pandas baru dengan nama "df." Ini memungkinkan Anda untuk mengakses, memanipulasi, dan menganalisis data dengan bantuan berbagai fitur yang disediakan oleh Pandas. Setelah itu, data ditampilkan **dengan print(df)**

Output :

```
        CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  \
0    0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900    1  296
1    0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622    3  222
..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...
501  0.06263   0.0  11.93   0.0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93   0.0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93   0.0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93   0.0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93   0.0  0.573  6.030   NaN  2.5050    1  273

     PTRATIO       B  LSTAT  MEDV
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       17.8  392.83   4.03  34.7
3       18.7  394.63   2.94  33.4
4       18.7  396.90    NaN  36.2
..       ...     ...    ...   ...
501     21.0  391.99    NaN  22.4
502     21.0  396.90   9.08  20.6
503     21.0  396.90   5.64  23.9
504     21.0  393.45   6.48  22.0
505     21.0  396.90   7.88  11.9

[506 rows x 14 columns]
```

Dataset perumahan Boston berisi 506 observasi dan 14 variabel yang terdiri dari :

- CRIM : Tingkat kejahatan per kapita menurut kota
- ZN : Proporsi lahan perumahan yang dikategorikan untuk lahan seluas lebih dari 25.000 kaki persegi.
- INDUS : Rasio Luas Lahan
- CHAS : Variabel dummy Sungai Charles (1 jika saluran membatasi sungai; 0 jika tidak)
- NOX : Konsentrasi oksida nitrat (bagian per 10 juta)
- RM : Jumlah rata-rata kamar per rumah
- AGE : Proporsi rumah yang dibangun sebelum tahun 1940
- DIS : Jarak ke pusat kerja Boston
- RAD : Indeks aksesibilitas ke jalan raya
- TAX : Tarif pajak properti nilai penuh per $10.000
- PTRATIO : Rasio murid-guru menurut kota
- B : 1000(Bk - 0.63)^2 rasio murid-guru menurut kota
- LSTAT : % penduduk berstatus rendah
- MEDV : Harga rumah

3. Mendeteksi Missing Value

```
[ ]  missing_value = df.isna().sum()
     print(missing_value)
```

Menghitung jumlah nilai yang hilang (missing values) dalam setiap kolom DataFrame Pandas "df" dan kemudian mencetak jumlah nilai yang hilang

Output :

```
CRIM       20
ZN         20
INDUS      20
CHAS       20
NOX         0
RM          0
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```

Output menunjukkan bahwa terdapat missing value pada kolom CRIM, ZN, INDUS, CHAS, AGE, dan LSTAT

4. Penanganan Missing Value

```
[ ]  # Menghapus missing value
     df.dropna(axis=0, inplace=True)

     # Mengecek kembali missing values
     missing_values_cleaned = df.isna().sum()
     print("\nJumlah Missing Values setelah Pembersihan:")
     print(missing_values_cleaned)
```

Menghapus baris-baris (axis=0) yang memiliki nilai yang hilang (missing values) dari DataFrame "df". Penghapusan dilakukan dengan menggunakan method **dropna()** dengan parameter **inplace=True**, yang berarti perubahan akan diterapkan secara langsung pada DataFrame "df" tanpa perlu menugaskan hasilnya ke DataFrame baru

Output :

```
Jumlah Missing Values setelah Pembersihan:
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

Terlihat bahwa sudah tidak ada missing value pada data

5. Menghapus kolom yang bukan numerik

```
# Menghapus kolom yang bukan numerik
df = df.drop(columns=['ZN','CHAS', 'B'])
df = df.sum()
df.head(12)
```

Menghapus kolom-kolom yang bukan merupakan kolom berjenis data numerik (misalnya, kolom dengan tipe data string atau kategori) dari DataFrame "df" agar data dapat divisualisasikan. Kemudian, menghitung total (jumlah) dari setiap kolom numerik yang tersisa setelah menghapus kolom-kolom tersebut.

Setelah dilakukan pengamatan pada jenis nilai datanya, data yang bukan merupakan numerik adalah kolom data ZN (Proporsi lahan perumahan yang dikategorikan untuk lahan seluas lebih dari 25.000 kaki persegi.), CHAS (Variabel dummy Sungai Charles (1 jika saluran membatasi sungai; 0 jika tidak), dan B (1000(Bk - 0.63)^2 rasio murid-guru menurut kota)

Output :

```
CRIM         1453.91365
INDUS        4334.34000
NOX           217.96690
RM           2474.32600
AGE         27159.50000
DIS          1499.27560
RAD          3705.00000
TAX        160134.00000
PTRATIO      7303.80000
LSTAT        5031.03000
MEDV         8809.70000
dtype: float64
```

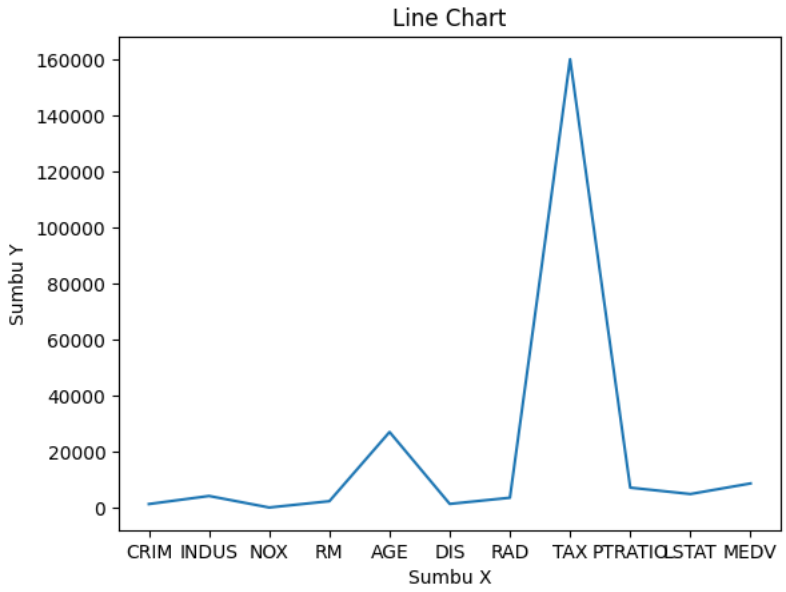
6. Visualisasi dengan Line Chart

Memvisualisasikan datang menggunakan diagram garis (line chart) menggunakan indeks DataFrame ("df") sebagai sumbu x dan nilai dari DataFrame tersebut sebagai sumbu y

```
[ ]  x= df.index
     y= df
     plt.plot(x, y)
     plt.title('Line Chart')
     plt.xlabel('Sumbu X')
     plt.ylabel('Sumbu Y')
     plt.show()
```

- Menentukan variabel **x** untuk menyimpan indeks dari DataFrame "df". Ini akan menjadi sumbu x dalam diagram garis.
- Variabel **y** digunakan untuk menyimpan seluruh DataFrame "df". Ini akan menjadi sumbu y dalam diagram garis.
- **plt.plot(x, y)** untuk membuat diagram garis dengan indeks sebagai sumbu x dan nilai dari DataFrame sebagai sumbu y.
- Memberi judul pada diagram menggunakan **plt.title()**.
- Memberi label sumbu x dan sumbu y dengan **plt.xlabel()** dan **plt.ylabel()**
- Menampilkan diagram garis dengan **plt.show()**.

Output :



Output menunjukkan bahwa sumbu Y dari diagram garis di atas adalah nilai dari variable. Sumbu X merupakan variabel data. Dari visualisasi diatas, dapat dilihat bahwa variable TAX memiliki nilai tertinggi. Sedangkan untuk atribut NOX memiliki nilai yang terendah.

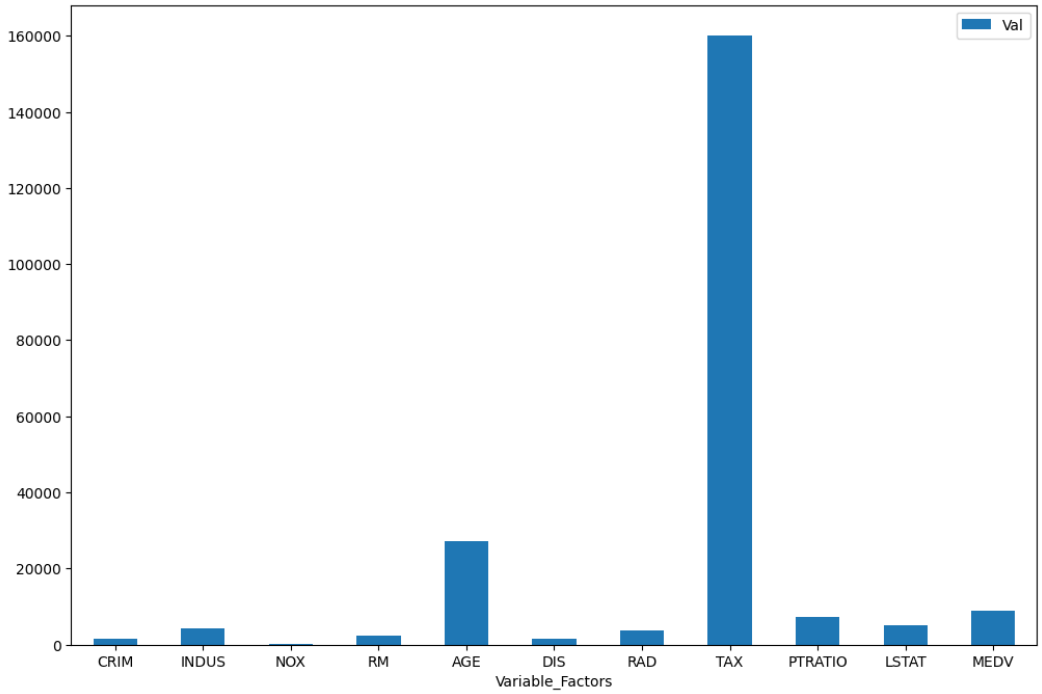
7. Visualisasi dengan Bar Chart

Memvisualisasikan datang menggunakan diagram batang (bar chart) dengan menggunakan DataFrame "df". Dalam diagram ini, sumbu x akan berisi variabel faktor dari DataFrame "df" dan sumbu y akan berisi nilai dari DataFrame tersebut.

```
[ ]  df = pd.DataFrame({'Variable_Factors':df.index, 'Val':df})
     ax = df.plot.bar(x='Variable_Factors', y='Val', rot=0, figsize=(12,8))
```

- **df = pd.DataFrame({'Variable_Factors':df.index, 'Val':df})** akan membuat sebuah DataFrame baru dengan menggunakan library pandas
- **x='Variable_Factors'** dan **y='Val'** untuk menentukan data yang akan digunakan untuk sumbu-x ("Variable_Factors") dan sumbu-y ("Val") dalam plot batang.
- **rot=0** untuk mengatur rotasi label sumbu-x menjadi 0 derajat, yang berarti label sumbu-x akan ditampilkan secara horizontal.
- **figsize=(12,8)** untuk mengatur ukuran gambar (plot) yang akan dihasilkan, dengan lebar 12 unit dan tinggi 8 unit.

Output :



Output menunjukkan bahwa sumbu Y dari diagram garis di atas adalah nilai dari variable. Sumbu X merupakan variabel data. Dari visualisasi diatas, dapat dilihat bahwa variable TAX memiliki nilai tertinggi. Sedangkan untuk atribut NOX memiliki nilai yang terendah.

8. Visualisasi dengan Pie Chart

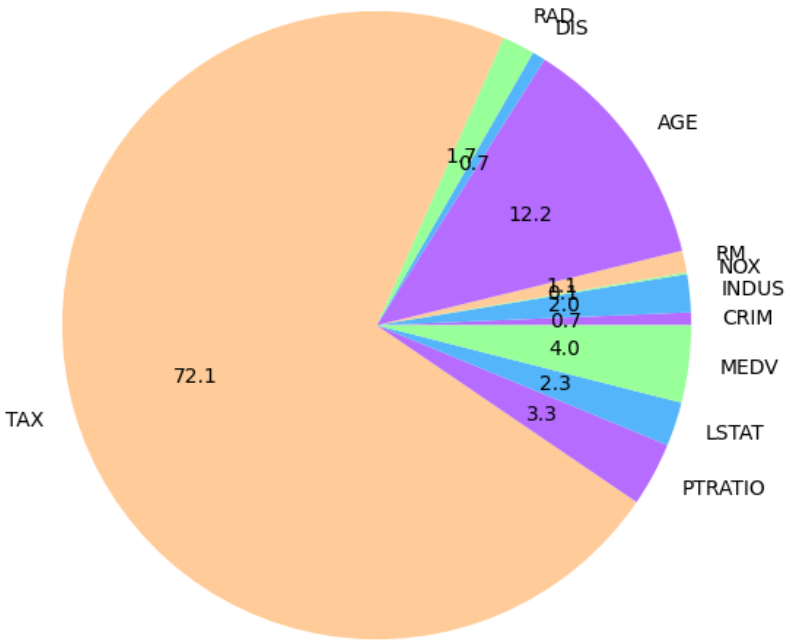
Pertama, **import matplotlib.pyplot as plt** terlebih dahulu untuk membuat visualisasi data seperti grafik, diagram, dan plot dalam Python.

```
[ ]  import matplotlib.pyplot as plt

     colors = ['#B66DFF','#54B5FB','#99ff99','#ffcc99']
     plt.figure(figsize = (7, 7))
     plt.pie(df['Val'], labels = df['Variable_Factors'], autopct = '%.1f', colors=colors)
     plt.show()
```

- **colors = ['#B66DFF','#54B5FB','#99ff99','#ffcc99']**: Mendefinisikan daftar warna yang akan digunakan untuk mengisi bagian-bagian diagram pie.
- **plt.figure(figsize=(7, 7))**: Mengatur ukuran gambar (figure) menjadi 7 unit lebar dan 7 unit tinggi.
- **plt.pie(df['Val'], labels=df['Variable_Factors'], autopct='%.1f', colors=colors)**: Membuat diagram pie.
  - **df['Val']**: Untuk menggambarkan diagram pie dari kolom "Val" dalam DataFrame "df".
  - **labels=df['Variable_Factors' :** Label yang ditampilkan untuk setiap bagian dalam diagram pie
  - **autopct='%.1f'** : Mengatur nilai yang ditampilkan pada diagram menjadi 1 angka decimal di belakang koma
  - **colors=colors**: Mengisi bagian-bagian dalam diagram pie sesuai urutan warna yang ditentukan.

Output :



Pada grafik diatas, variable TAX memiliki bagian yang terluas, yaitu 72,1% dari seluruh bagian. Sedangkan, NOX hanya menempati 0,1% bagian dari grafik.

9. Visualisasi dengan Area Chart

- Melakukan persiapan data kembali sebelum membuat diagram area agar

```
[ ]  data = pd.read_csv("HousingData.csv")
     df =  pd.DataFrame(data)
     df.isna().sum()
     df = df.fillna(method='ffill')
     kolom_yang_akan_dihapus = ['ZN','CHAS', 'B']
     df = df.drop(kolom_yang_akan_dihapus, axis=1)
     print(df.head(8))
```

Output :

```
      CRIM  INDUS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  LSTAT  MEDV
0  0.00632   2.31  0.538  6.575  65.2  4.0900    1  296     15.3   4.98  24.0
1  0.02731   7.07  0.469  6.421  78.9  4.9671    2  242     17.8   9.14  21.6
2  0.02729   7.07  0.469  7.185  61.1  4.9671    2  242     17.8   4.03  34.7
3  0.03237   2.18  0.458  6.998  45.8  6.0622    3  222     18.7   2.94  33.4
4  0.06905   2.18  0.458  7.147  54.2  6.0622    3  222     18.7   2.94  36.2
5  0.02985   2.18  0.458  6.430  58.7  6.0622    3  222     18.7   5.21  28.7
6  0.08829   7.87  0.524  6.012  66.6  5.5605    5  311     15.2  12.43  22.9
7  0.14455   7.87  0.524  6.172  96.1  5.9505    5  311     15.2  19.15  27.1
```

- **B**

```
[ ]  # Membuat diagram area
     df.plot(kind='area')

     # Memberi judul dan label sumbu
     plt.title('Data Perumahan Boston')
     plt.ylabel('Faktor yang Mempengaruhi')
     plt.xlabel('Harga Rumah')

     # Menampilkan diagram
     plt.show()
```

Output :