```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns


url = "https://raw.githubusercontent.com/tidyverse/ggplot2/main/data-raw/diamonds.csv"
diamons = pd.read_csv(url)
diamons
```

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...
53935	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.50
53936	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61
53937	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56
53938	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74
53939	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64

53940 rows × 10 columns

Exploratory Data Analysis - Deskripsi Variabel

- Harga dalam dolar Amerika Serikat ($) adalah fitur target.
- carat: merepresentasikan bobot (weight) dari diamonds (0.2-5.01), digunakan sebagai ukuran dari batu permata dan perhiasan.
- cut: merepresentasikan kualitas pemotongan diamonds (Fair, Good, Very Good, Premium, and Ideal).
- color: merepresentasikan warna, dari J (paling buruk) ke D (yang terbaik).
- clarity: merepresentasikan seberapa jernih diamonds (I1 (paling buruk), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (terbaik))
- x: merepresentasikan panjang diamonds dalam mm (0-10.74).
- y: merepresentasikan lebar diamonds dalam mm (0-58.9).
- z: merepresentasikan kedalaman diamonds dalam mm (0-31.8).
- depth: merepresentasikan z/mean(x, y) = 2 * z/(x + y) (43-79).
- table: merepresentasikan lebar bagian atas berlian relatif terhadap titik terlebar 43-95).

```
diamons.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53940 entries, 0 to 53939
Data columns (total 10 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   carat    53940 non-null  float64
 1   cut      53940 non-null  object 
 2   color    53940 non-null  object 
 3   clarity  53940 non-null  object 
 4   depth    53940 non-null  float64
 5   table    53940 non-null  float64
 6   price    53940 non-null  int64  
 7   x        53940 non-null  float64
 8   y        53940 non-null  float64
 9   z        53940 non-null  float64
dtypes: float64(6), int64(1), object(3)
memory usage: 4.1+ MB
```

- terdapat 3 kolom dengan tipe object (non numeric)
- terdapat 6 kolom dengan tipe float (numeric)
- terdapat 1 kolom numerik dengan tipe data int

```
diamons.describe()
```

	carat	depth	table	price	x	
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.0
mean	0.797940	61.749405	57.457184	3932.799722	5.731157	5.7
std	0.474011	1.432621	2.234491	3989.439738	1.121761	1.1
min	0.200000	43.000000	43.000000	326.000000	0.000000	0.0
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.7
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.7
75%	1.040000	62.500000	59.000000	5324.250000	6.540000	6.5
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.9

Missing Value

```
x = (diamons.x == 0).sum()
y = (diamons.y == 0).sum()
z = (diamons.z == 0).sum()

print('Nilai 0 di kolom x ada : ', x)
print('Nilai 0 di kolom y ada : ', y)
print('Nilai 0 di kolom z ada : ', z)
```

```
Nilai 0 di kolom x ada :  8
Nilai 0 di kolom y ada :  7
Nilai 0 di kolom z ada :  20
```

```
diamons.loc[(diamons['z'] == 0)]
```

	carat	cut	color	clarity	depth	table	price	x	y	z
2207	1.00	Premium	G	SI2	59.1	59.0	3142	6.55	6.48	0.0
2314	1.01	Premium	H	I1	58.1	59.0	3167	6.66	6.60	0.0
4791	1.10	Premium	G	SI2	63.0	59.0	3696	6.50	6.47	0.0
5471	1.01	Premium	F	SI2	59.2	58.0	3837	6.50	6.47	0.0
10167	1.50	Good	G	I1	64.0	61.0	4731	7.15	7.04	0.0
11182	1.07	Ideal	F	SI2	61.6	56.0	4954	0.00	6.62	0.0
11963	1.00	Very Good	H	VS2	63.3	53.0	5139	0.00	0.00	0.0
13601	1.15	Ideal	G	VS2	59.2	56.0	5564	6.88	6.83	0.0
15951	1.14	Fair	G	VS1	57.5	67.0	6381	0.00	0.00	0.0
24394	2.18	Premium	H	SI2	59.4	61.0	12631	8.49	8.45	0.0
24520	1.56	Ideal	G	VS2	62.2	54.0	12800	0.00	0.00	0.0
26123	2.25	Premium	I	SI1	61.3	58.0	15397	8.52	8.42	0.0
26243	1.20	Premium	D	VVS1	62.1	59.0	15686	0.00	0.00	0.0
27112	2.20	Premium	H	SI1	61.2	59.0	17265	8.42	8.37	0.0
27429	2.25	Premium	H	SI2	62.8	59.0	18034	0.00	0.00	0.0
27503	2.02	Premium	H	VS2	62.7	53.0	18207	8.02	7.95	0.0
27739	2.80	Good	G	SI2	63.8	58.0	18788	8.90	8.85	0.0
49556	0.71	Good	F	SI2	64.1	60.0	2130	0.00	0.00	0.0
49557	0.71	Good	F	SI2	64.1	60.0	2130	0.00	0.00	0.0
51506	1.12	Premium	G	I1	60.4	59.0	2383	6.71	6.67	0.0

```
diamons = diamons.loc[(diamons[['x','y','z']]!=0).all(axis=1)]
```

```
diamons.shape
```

```
(53920, 10)
```

```
diamons.describe()
```

	carat	depth	table	price	x	
count	53920.000000	53920.000000	53920.000000	53920.000000	53920.000000	53920.0
mean	0.797698	61.749514	57.456834	3930.993231	5.731627	5.7
std	0.473795	1.432331	2.234064	3987.280446	1.119423	1.1
min	0.200000	43.000000	43.000000	326.000000	3.730000	3.6
25%	0.400000	61.000000	56.000000	949.000000	4.710000	4.7
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.7
75%	1.040000	62.500000	59.000000	5323.250000	6.540000	6.5
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.9

Menangani Outliers

- teknik menangani ouliers

1. Hypothesis Testing
2. Z-score method
3. IQR Method

- IQR

```
Q1 = diamons.quantile(0.25)
Q3 = diamons.quantile(0.75)
IQR = Q3 - Q1
diamons = diamons[~((diamons < (Q1 - 1.5 * IQR))|(diamons > (Q3 + 1.5 * IQR))).any(axis=1)]

# cek ukuran dataset setelah kita drop duplicaters
diamons.shape
```

```
<ipython-input-80-5140a843280b>:1: FutureWarning: The default value of numeric_only in DataFrame.quantile is deprecated. In a futur
  Q1 = diamons.quantile(0.25)
<ipython-input-80-5140a843280b>:2: FutureWarning: The default value of numeric_only in DataFrame.quantile is deprecated. In a futur
  Q3 = diamons.quantile(0.75)
<ipython-input-80-5140a843280b>:4: FutureWarning: Automatic reindexing on DataFrame vs Series comparisons is deprecated and will ra
  diamons = diamons[~((diamons < (Q1 - 1.5 * IQR))|(diamons > (Q3 + 1.5 * IQR))).any(axis=1)]
(47524, 10)
```

EDA (Explorati Data Analys)

melakukan analysis

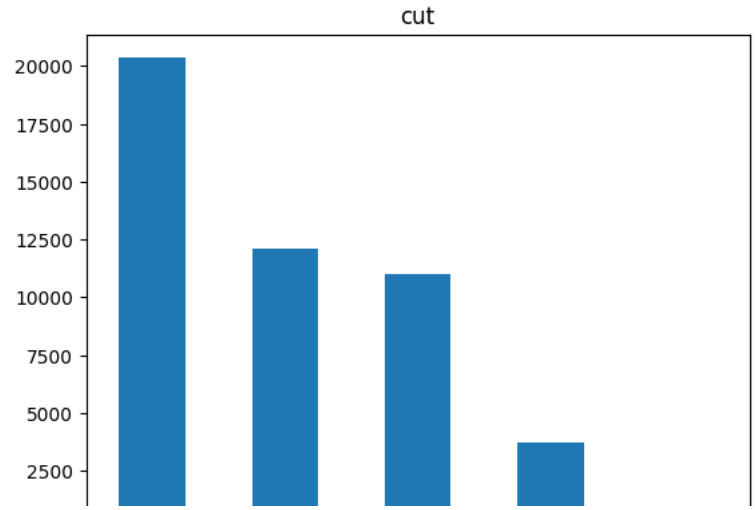
Univariate EDA

```
numerical_features = ['price','carat','depth','table','x','y','z']
categorical_features = ['cut','color','clarity']
```

feature cut

```
feature = categorical_features[0]
count = diamons[feature].value_counts()
percent = 100*diamons[feature].value_counts(normalize=True)
df = pd.DataFrame({'jumlah sample': count, 'persentase' : percent.round(1)})
print(df)

count.plot(kind='bar', title=feature)
plt.show()
```

	jumlah sample	persentase
Ideal	20340	42.8
Premium	12115	25.5
Very Good	10994	23.1
Good	3745	7.9
Fair	330	0.7



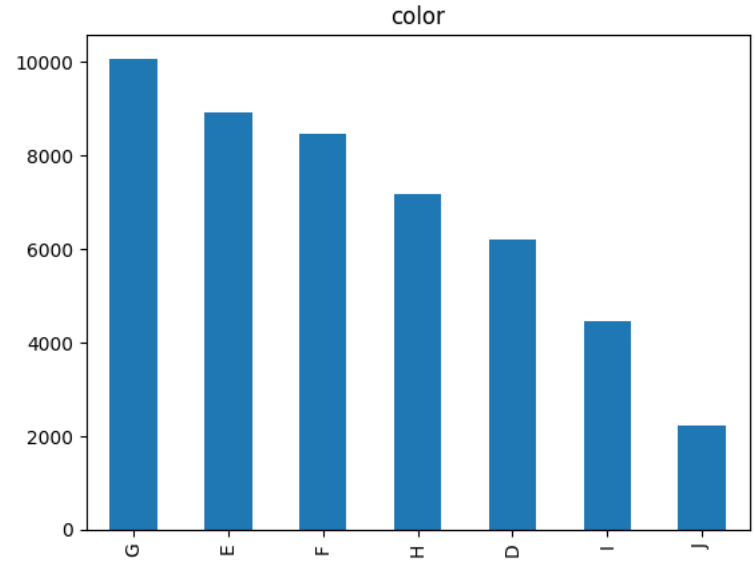
dari hasil visualisasi diatas terdapat 5 kategori pada fitur cut, yaitu ideal, premium, very good, good, fair. dari data presentase dapat kita simpulkan bahwa lebih dari 60% sampel merupakan diamonds tipe grade tinggi yaitu ideal dan premium

re ry

feature color

```
features = diamons.columns[2]
count = diamons[features].value_counts()
percent = 100*diamons[features].value_counts(normalize=True)
df = pd.DataFrame({'jumlah sample':count, 'percentase':percent.round(1)})
print(df)

count.plot(kind='bar', title=features)
plt.show()
```

	jumlah sample	percentase
G	10081	21.2
E	8910	18.7
F	8466	17.8
H	7176	15.1
D	6195	13.0
I	4462	9.4
J	2234	4.7

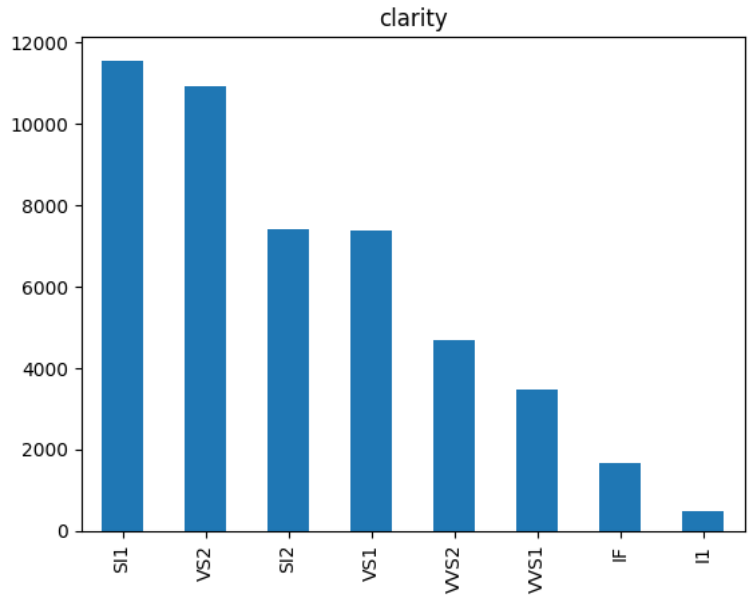


berdasarkan deskripsi variabel urutan warna dari yang paling buruk ke yang paling bagus adalah J,I,H,G,F,E dan D. dari grafik di atas dapat kita simpulkan bahwa sebagian besar grade berada pada grade menengah, yaitu grade G, F, H

```
feature = diamons.columns[3]
count = diamons[feature].value_counts()
percent = 100*diamons[feature].value_counts(normalize=True)
```

```
df = pd.DataFrame({'jumlah sample':count, 'percent':percent.round(1)})
print(df)

count.plot(kind='bar', title=feature)
plt.show()
```

```
      jumlah sample  percent
SI1           11552     24.3
VS2           10928     23.0
SI2            7402     15.6
VS1            7373     15.5
VVS2           4683      9.9
VVS1           3463      7.3
IF             1650      3.5
I1              473      1.0
```



berdasarkan dari deskripsi variabel fiture clarity terdiri dari 8 kategori dari yang paling buruk ke yang paling baik yaitu: I1, SI2, SI1, VS2,VS1,VVS2,VVS1,IF

'IF' - Internally Flawless

'VVS2' - Very Very Slight Inclusions

'VVS1' - Very Very Slight Inclusions

'VS1' - Very Slight Inclusions

'VS2' - Very Slight Inclusions

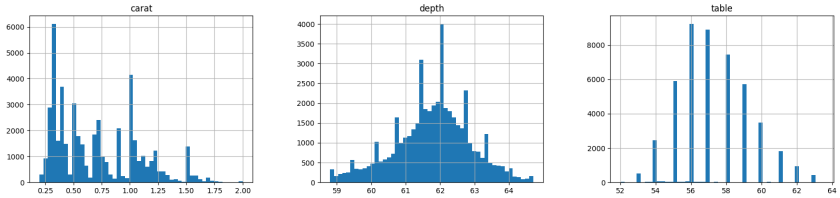
'SI2' - Slight Inclusions

'SI1' - Slight Inclusions

'I1' - Imperfect

NUMERIC FEATURES

```
diamons.hist(bins=50, figsize=(20, 15))
plt.show()
```



kita amati histogram 'price' yang akan menjadi target kita kita bisa memperoleh beberapa informasi, diantaranya :

- peningkatan diamons sebanding dengan penurunan jumlah sampel. hal ini dapat kita lihat jelas dari histogram 'price' yang grafiknya mengalami penurunan seiring dengan semakin banyaknya jumlah sampel (sumbu x)
- rendang harga diamons cukup tinggi yaitu dari skala ratusan dolar amerika hingga sekitar $11800
- setengah harga berilian dibawah $2500
- distribusi harga miring ke kanan (right-skewed) hal ini akan berimplikasi pada model

EDA - Multivariate Analys

Categorical Features

```
cat_feature = diamons.select_dtypes(include='object').columns.to_list()


for col in cat_feature:
  sns.catplot(x=col, y='price', kind='bar', dodge=False, height=4, aspect=3, data=diamons, palette='Set3')
  plt.title('Rata-rata "price" relatif terhadap - {}'.format(col))
```

dengan mengamati rata-rata harga relatif terhadap fitur kategori di atas, kita memperoleh insight sebagai berikut:

- pada fitur 'cut', rata-rata harga cenderung mirip. rentangnya berada antara 3500 hingga 4500. grade tertinggi yaitu grade ideal memiliki harga rata-rata terendah diantara grade lainnya. sehingga, fitur cut memiliki pengaruh atau dampak yang kecil terhadap rata-rata harga.
- pada fitur 'color' semakin rendah grade warna, harga diamonds justru semakin tinggi. dari sini dapat disimpulkan bawa warna memiliki pengaruh yang rendah terhadap harga.
- pada fitur 'clarity', secara umum, diamond dengan grade lebih rendah memiliki harga yang lebih tinggi. hal ini berarti bahwa fitur 'clarity' memiliki pengaruh yang rendah terhadap harga.

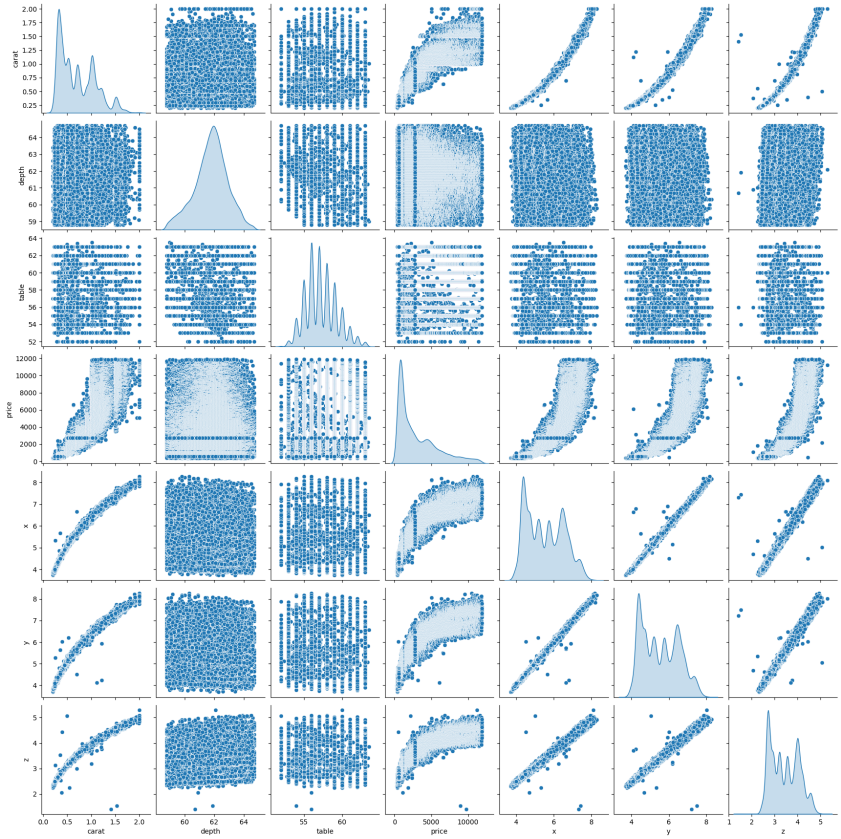
Kesimpulan akhir, fitur kategori memiliki pengaruh yang rendah terhadap harga

Numerical feature

mengamati hubungan antara firut numerik menggunakan fungsi pairplot()

```
# mengamati hubungan antara fituur numerik dengan fungsi pairplot()
sns.pairplot(diamons, diag_kind= 'kde')
```

```
<seaborn.axisgrid.PairGrid at 0x7f26eb4d8280>
```




```
plt.figure(figsize=(10, 8))
correlation_matrix = diamons.corr().round(2)

# untuk menge-print nilai dalam kotak, menggunakan parameter anot=True
sns.heatmap(data=correlation_matrix, annot=True, cmap='coolwarm', linewidth=0.5)
plt.title('Correlation Matrix untuk Fitur Numerik', size=20)
plt.show()
```

```
<ipython-input-89-88d1d3a362f4>:2: FutureWarning: The default value of numeric_onl
  correlation_matrix = diamons.corr().round(2)
```


```
diamons.drop(['depth'], inplace=True, axis=1)
diamons.head()
```

	carat	cut	color	clarity	table	price	x	y	z
0	0.23	Ideal	E	SI2	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	61.0	326	3.89	3.84	2.31
3	0.29	Premium	I	VS2	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	58.0	335	4.34	4.35	2.75
5	0.24	Very Good	J	VVS2	57.0	336	3.94	3.96	2.48

Data Preparation

```
# melakukan one-hot-encoding untuk fitur yang bertipe kategori

from sklearn.preprocessing import OneHotEncoder

diamons = pd.concat([diamons, pd.get_dummies(diamons['cut'], prefix='cut')], axis=1)
diamons = pd.concat([diamons, pd.get_dummies(diamons['color'], prefix='color')], axis=1)
diamons = pd.concat([diamons, pd.get_dummies(diamons['clarity'], prefix='clarity')], axis=1)

diamons.drop(['cut','color','clarity'], axis=1, inplace=True)
diamons.head()
```

	carat	table	price	x	y	z	cut_Fair	cut_Good	cut_Ideal	cut_Premium
0	0.23	55.0	326	3.95	3.98	2.43	0	0	1	0
1	0.21	61.0	326	3.89	3.84	2.31	0	0	0	1
3	0.29	58.0	334	4.20	4.23	2.63	0	0	0	1
4	0.31	58.0	335	4.34	4.35	2.75	0	1	0	0
5	0.24	57.0	336	3.94	3.96	2.48	0	0	0	0

5 rows × 26 columns

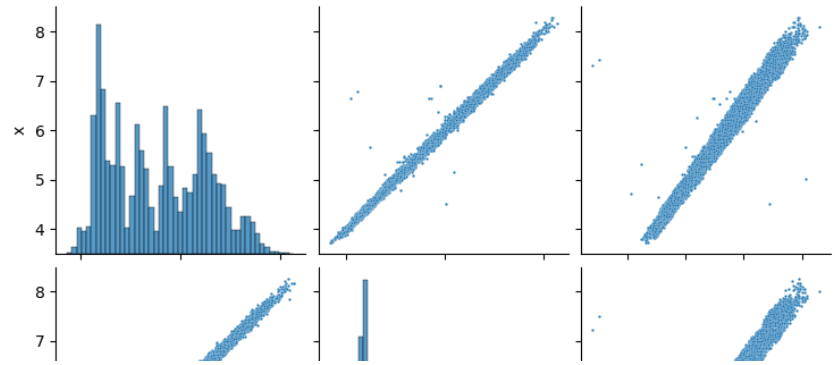
reduksi dimensi dengan PCA

Principal Component Analisis

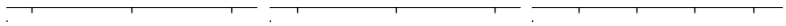
PAC adalah teknik untuk meredudansi dimensi, mengekstraksi fitur dan mentransformasi data dari "n-dimensional space" ke dalam sistem berkoordinat baru dengan dimensi m

```
sns.pairplot(diamons[['x','y','z']], plot_kws={'s':3})
```

```
<seaborn.axisgrid.PairGrid at 0x7f26eb07b760>
```




```
from sklearn.decomposition import PCA

pca = PCA(n_components=3, random_state=123)
pca.fit(diamons[['x','y','z']])
princ_comp = pca.transform(diamons[['x','y','z']])
```



```
princ_comp
```

```
array([[-2.45258471,  0.01805597, -0.01052123],
       [-2.63006596,  0.06997942,  0.04782566],
       [-2.04852967, -0.02359652, -0.01266468],
       ...,
       [ 0.20939029, -0.05183578, -0.01258338],
       [ 0.8841492 ,  0.04570632,  0.02327686],
       [ 0.47464307, -0.0228551 , -0.02676376]])
```



```
pca.explained_variance_ratio_.round(3)
```

```
array([0.998, 0.002, 0.001])
```

hasil dari output di atas, 99.8% informasi pada ketiga fitur 'x','y','z' terdapat pada pc pertama. sedangkan sisanya, sebesar 0.2% dan 0.1% terdapat pada pc kedua dan ketiga.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=1, random_state=123)
pca.fit(diamons[['x','y','z']])
diamons['dimension'] = pca.transform(diamons.loc[:, ('x','y','z')]).flatten()
diamons.drop(['x','y','z'], inplace=True, axis=1)
```

```
diamons.head()
```

	carat	table	price	cut_Fair	cut_Good	cut_Ideal	cut_Premium	cut_Very Good	color
0	0.23	55.0	326	0	0	1	0	0	
1	0.21	61.0	326	0	0	0	1	0	
3	0.29	58.0	334	0	0	0	1	0	
4	0.31	58.0	335	0	1	0	0	0	
5	0.24	57.0	336	0	0	0	0	1	

5 rows × 24 columns

train test split

jika data berjumlah kecil maka pembagian 80:20 sangat ideal

jika data berjumlah besar seperti 5juta maka gunakan 90:10 untuk data uji 10 dan data latih 90

```
from sklearn.model_selection import train_test_split

x = diamons.drop(['price'], axis=1)
y = diamons.price
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=123, test_size=0.1)
```

```
print(f'Total # of sample in whole dataset: {len(x)}')
print(f'Total # of sample in train dataset: {len(x_train)}')
print(f'Total # of sample in test dataset: {len(x_test)}')
```

```
Total # of sample in whole dataset: 47524
Total # of sample in train dataset: 42771
Total # of sample in test dataset: 4753
```

```
from sklearn.preprocessing import StandardScaler

numerical_features = ['carat', 'table', 'dimension']
scaler = StandardScaler()
scaler.fit(x_train[numerical_features])
x_train[numerical_features] = scaler.transform(x_train.loc[:, numerical_features])
x_train[numerical_features].head()
```

	carat	table	dimension
536	-0.026226	0.864091	0.143464
21293	1.348407	1.359644	1.353588
45577	-0.511390	-0.622566	-0.372761
37379	-0.834833	-0.622566	-0.905790
38240	-0.861787	-0.622566	-0.813165

```
x_train[numerical_features].describe().round(4)
```

	carat	table	dimension
count	42771.0000	42771.0000	42771.0000
mean	0.0000	-0.0000	-0.0000
std	1.0000	1.0000	1.0000
min	-1.3739	-2.6048	-1.8867
25%	-0.8887	-0.6226	-0.9283
50%	-0.2688	-0.1270	-0.1063
75%	0.8093	0.8641	0.8847
max	3.4777	3.0941	2.6998

Model Development

```
models = pd.DataFrame(index=['train_mse', 'test_mse'],
                      columns=['KNN', 'RandomForest', 'Boosting'])
```

```
# K-Nearst Neighbor (classification / regression)
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
```

```
knn = KNeighborsRegressor(n_neighbors=10)
knn.fit(x_train, y_train)

models.loc['train_mse','KNN'] = mean_squared_error(y_pred = knn.predict(x_train), y_true=y_train)
```

```
models
```

	KNN	RandomForest	Boosting
train_mse	203761.113997	NaN	NaN
test_mse	NaN	NaN	NaN

```
# RandomForestRegressor (classifikation / regression)
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n_estimators=50, max_depth=16, random_state=55, n_jobs=-1)
rf.fit(x_train, y_train)
```

```
models.loc['train_mse', 'RandomForest'] = mean_squared_error(y_pred = rf.predict(x_train), y_true=y_train)
```

```
models
```

	KNN	RandomForest	Boosting
train_mse	203761.113997	52287.365706	NaN
test_mse	NaN	NaN	NaN

```
# boosting (regression)
## Adaptive boosting
## Gradient boosting

# adaptive boosting (-)
from sklearn.ensemble import AdaBoostRegressor

boosting = AdaBoostRegressor(learning_rate=0.05, random_state=55)
boosting.fit(x_train, y_train)
models.loc['train_mse', 'Boosting'] = mean_squared_error(y_pred = boosting.predict(x_train), y_true = y_train )
```

```
models
```

	KNN	RandomForest	Boosting
train_mse	203761.113997	52287.365706	904838.012908
test_mse	NaN	NaN	NaN

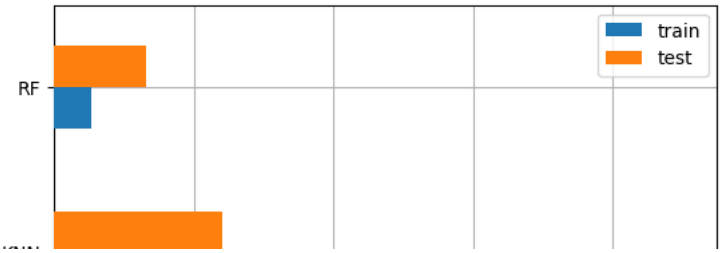
Evaluation Model

```
# scaling befor evaluation
# lakukan scaling pada feature numeric pada x_test sehingga miliki rata-rata=0 dan varians=1

x_test.loc[:, numerical_features] = scaler.transform(x_test[numerical_features])
```

```
# melakukan evaluasi
# buat variable mse yang isinya adalh dataframe nilai mse data train dan data test pada masing-masing algoritma

mse = pd.DataFrame(columns=['train', 'test'], index=['KNN','RF','Boosting'])

# buat dictionary untuk setiapp algoritma yang digunakan
model_dict = {'KNN':knn, 'RF':rf, 'Boosting':boosting}

# hitung MSE masing-masing algoritma pada data train dan test
for name, model in model_dict.items():
  mse.loc[name, 'train'] = mean_squared_error(y_true = y_train, y_pred = model.predict(x_train))/1e3
  mse.loc[name, 'test'] = mean_squared_error(y_true = y_test, y_pred = model.predict(x_test))/1e3

mse
```

	train	test
KNN	203.761114	239.529923
RF	52.287366	130.788418
Boosting	904.838013	846.212966

```
fix, ax = plt.subplots()

mse.sort_values(by='test', ascending=False).plot(kind='barh', ax=ax, zorder=3)
ax.grid(zorder=0)
```




```
# ujicoba harga diamond dengan algoritma yang memiliki error kecil

prediksi = x_test.iloc[:2].copy()
pred_dict = {'y_true':y_test[:2]}

for name, model in model_dict.items():
  pred_dict['prediksi_'+name] = model.predict(prediksi).round(1)

pd.DataFrame(pred_dict)
```

	y_true	prediksi_KNN	prediksi_RF	prediksi_Boosting
35096	886	923.2	884.2	788.1
17479	7018	6044.8	7674.8	9178.2

random forest memberikan hasil yang paling mendekati nilai sungguh

```
prediksi
```

	carat	table	cut_Fair	cut_Good	cut_Ideal	cut_Premium	cut_Very Good	c
35096	-1.104369	-1.613671	0	0	1	0	0	
17479	2.157014	2.350749	0	0	0	1	0	

2 rows × 23 columns