```
import pandas as pd
# read csv using pandas
data = pd.read_csv('https://raw.githubusercontent.com/pycaret/pycaret/master/datasets/diamond.csv')

# check the head of dataframe
data.head()
```

	Carat Weight	Cut	Color	Clarity	Polish	Symmetry	Report	Price
0	1.10	Ideal	H	SI1	VG	EX	GIA	5169
1	0.83	Ideal	H	VS1	ID	ID	AGSL	3470
2	0.85	Ideal	H	SI1	EX	EX	GIA	3183
3	0.91	Ideal	E	SI1	VG	VG	GIA	4370
4	0.83	Ideal	G	SI1	EX	EX	GIA	3171

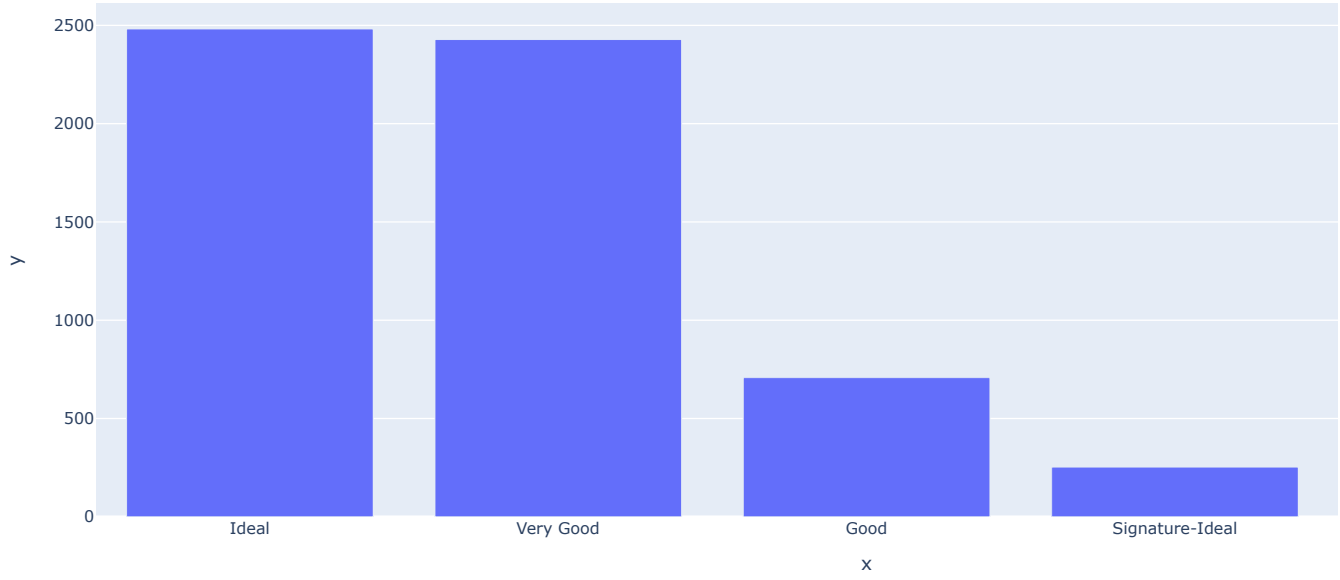
```
# check the data types
data.dtypes
```

```
Carat Weight    float64
Cut              object
Color            object
Clarity          object
Polish           object
Symmetry         object
Report           object
Price             int64
dtype: object
```

```
# read csv using pandas
import pandas as pd
data = pd.read_csv('https://raw.githubusercontent.com/pycaret/pycaret/master/datasets/diamond.csv')

# check value counts of Cut column
data['Cut'].value_counts()
```

```
Ideal              2482
Very Good          2428
Good                708
Signature-Ideal     253
Fair                129
Name: Cut, dtype: int64
```

```
import plotly.express as px
cut_counts = data['Cut'].value_counts()
fig = px.bar(x=cut_counts.index, y=cut_counts.values)
fig.show()
```



groupby()` is a function in Pandas that allows you to group data by one or more columns and apply aggregate functions such as sum, mean, and count.

```
# read csv using pandas
import pandas as pd
data = pd.read_csv('https://raw.githubusercontent.com/pycaret/pycaret/master/datasets/diamond.csv')

# average carat weight and price by Cut
data.groupby(by = 'Cut').mean()
```

```
<ipython-input-8-a50f3bddfded>:6: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default
```

Cut	Carat Weight	Price
Fair	1.058682	5886.178295
Good	1.268927	9326.656780
Ideal	1.382293	13127.331185
Signature-Ideal	1.205217	11541.525692
Very Good	1.332941	11484.696870

`crosstab()` is a function in pandas that creates a cross-tabulation table, which shows the frequency distribution of two or more categorical variables. This function is useful when you want to see the relationship between two or more categorical variables, such as how the frequency of one variable is related to another variable.

```
# read csv using pandas
import pandas as pd
data = pd.read_csv('https://raw.githubusercontent.com/pycaret/pycaret/master/datasets/diamond.csv')

# cross tab of Cut and Color
pd.crosstab(index=data['Cut'], columns=data['Color'])
```

Color	D	E	F	G	H	I
Cut						
Fair	12	32	24	21	24	16
Good	74	110	133	148	128	115
Ideal	280	278	363	690	458	413
Signature-Ideal	30	35	38	64	45	41
Very Good	265	323	455	578	424	383

pivot_table()` is a function in Pandas that creates pivot tables, which are similar to cross-tabulation tables but with more flexibility. This function is useful when you want to analyze multiple categorical variables and their relationship to one or more numeric variables.

```
import numpy as np
# read csv using pandas
import pandas as pd
data = pd.read_csv('https://raw.githubusercontent.com/pycaret/pycaret/master/datasets/diamond.csv')

# create pivot table
pd.pivot_table(data, values='Price', index='Cut', columns='Color', aggfunc=np.mean)
```

Color	D	E	F	G	H	I
Cut						
Fair	6058.250000	5370.625000	6063.625000	7345.523810	5908.500000	4573.187500
Good	10058.716216	8969.545455	9274.007519	9988.614865	9535.132812	8174.113043
Ideal	18461.953571	12647.107914	14729.426997	13570.310145	11527.700873	9459.588378
Signature-Ideal	19823.100000	11261.914286	13247.947368	10248.296875	9112.688889	8823.463415
Very Good	13218.826415	12101.910217	12413.905495	12354.013841	10056.106132	8930.031332

In Label encoding, each label is converted into an integer value. We will create a variable that contains the categories representing the education qualification of a person.

```
! pip install --upgrade category_encoders

import sklearn
sklearn.set_config(transform_output="pandas")
#pip install category_encoders
import category_encoders as ce
import pandas as pd
train_df=pd.DataFrame({'Degree':['High school','Masters','Diploma','Bachelors','Bachelors','Masters','Phd','High school','Hi

# create object of Ordinalencoding
encoder= ce.OrdinalEncoder(cols=['Degree'],return_df=True,
                           mapping=[{'col':'Degree',
'mapping':{'None':0,'High school':1,'Diploma':2,'Bachelors':3,'Masters':4,'phd':5}}])

#Original data
print(train_df)
```

```
    Requirement already satisfied: category_encoders in /usr/local/lib/python3.10/dist-packages (2.6.3)
    Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.23.5
    Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders)
    Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.11.4)
    Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0
    Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.5.3)
    Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.5.4)
    Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->ca
    Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_enc
    Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.1->category_encoders) (1.
    Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->cate
    Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.
    Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.9.0->cate
           Degree
    0  High school
    1      Masters
    2      Diploma
    3    Bachelors
    4    Bachelors
    5      Masters
    6          Phd
    7  High school
    8  High school
```

```
df_train_transformed = encoder.fit_transform(train_df)
df_train_transformed
```

	Degree
0	1.0
1	4.0
2	2.0
3	3.0
4	3.0
5	4.0
6	-1.0
7	1.0
8	1.0

In one hot encoding, for each level of a categorical feature, we create a new variable. Each category is mapped with a binary variable containing either 0 or 1. Here, 0 represents the absence, and 1 represents the presence of that category.

```
#one hot encoding
import category_encoders as ce
import pandas as pd
data=pd.DataFrame({'City':[
'Delhi','Mumbai','Hydrabad','Chennai','Bangalore','Delhi','Hydrabad','Bangalore','Delhi'
]})

#Create object for one-hot encoding
encoder=ce.OneHotEncoder(cols='City',handle_unknown='return_nan',return_df=True,use_cat_names=True)

#Original Data
data
```

	City
0	Delhi
1	Mumbai
2	Hydrabad
3	Chennai
4	Bangalore
5	Delhi
6	Hydrabad
7	Bangalore
8	Delhi

```
data_encoded = encoder.fit_transform(data)
data_encoded
```

	City_Delhi	City_Mumbai	City_Hydrabad	City_Chennai	City_Bangalore
0	1.0	0.0	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0
4	0.0	0.0	0.0	0.0	1.0
5	1.0	0.0	0.0	0.0	0.0
6	0.0	0.0	1.0	0.0	0.0
7	0.0	0.0	0.0	0.0	1.0
8	1.0	0.0	0.0	0.0	0.0

```
import pandas as pd

# generate df with 1 col and 4 rows
data = {
    "fruit": ["apple", "banana", "orange", "apple"]
}

# show head
df = pd.DataFrame(data)
df.head()
```

	fruit
0	apple
1	banana
2	orange
3	apple

```
# apply get_dummies function
df_encoded = pd.get_dummies(df["fruit"])
df_encoded .head()
```

	apple	banana	orange
0	1	0	0
1	0	1	0
2	0	0	1
3	1	0	0

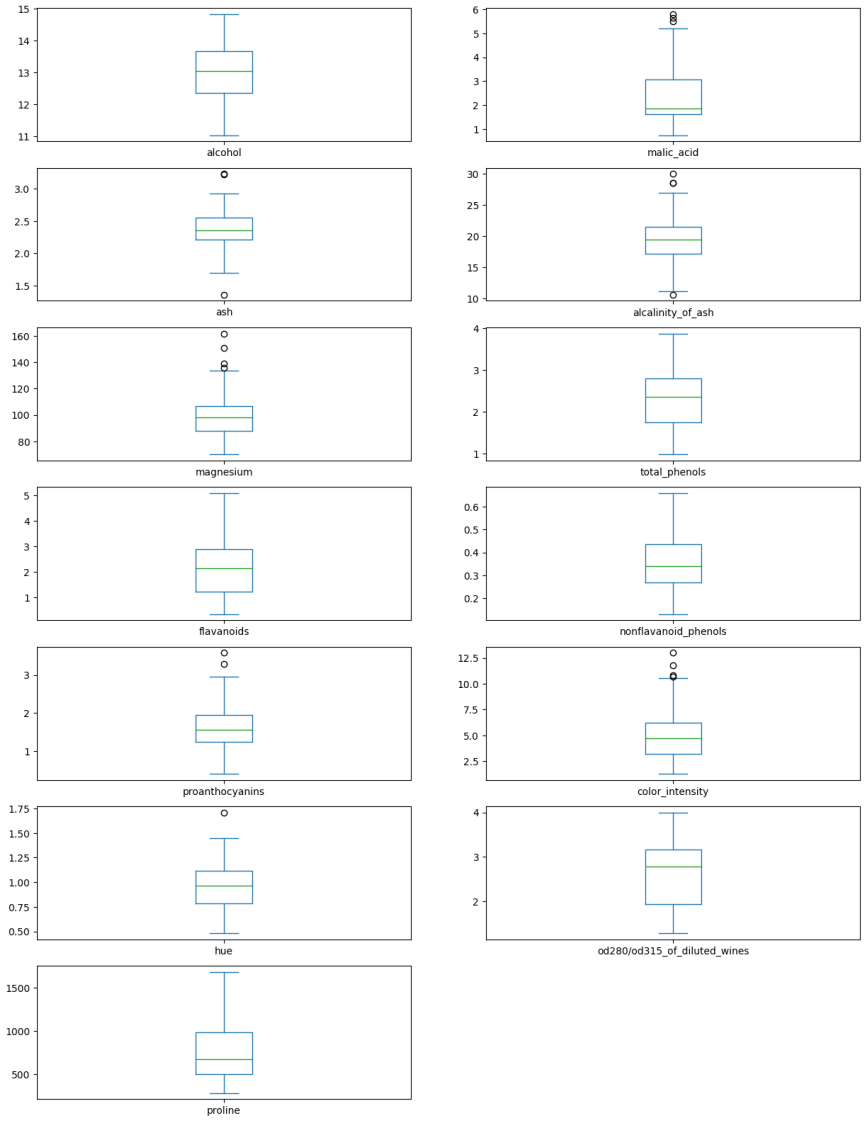
```
import pandas as pd

# generate df with 1 col and 4 rows
data = {
    "fruit": ["apple", "banana", "orange", "apple"]
}

# one-hot-encode using sklearn
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder()
encoded_results = encoder.fit_transform(df).toarray()
```

	fruit
0	apple
1	banana
2	orange
3	apple

```
from sklearn.datasets import load_wine
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.DataFrame(load_wine()["data"],columns=load_wine()["feature_names"])
data.head()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flava
0	14.23	1.71	2.43	15.6	127.0	2.80	
1	13.20	1.78	2.14	11.2	100.0	2.65	
2	13.16	2.36	2.67	18.6	101.0	2.80	
3	14.37	1.95	2.50	16.8	113.0	3.85	
4	13.24	2.59	2.87	21.0	118.0	2.80	

```
data.plot(kind="box",subplots=True,layout=(7,2),figsize=(15,20));
```


alcohol
malic_acid
ash
alcalinity_of_ash
magnesium
total_phenols
flavanoids
nonflavanoid_phenols
proanthocyanins
color_intensity
hue
od280/od315_of_diluted_wines
proline

```
import numpy as np

# Generate a random dataset with outliers
np.random.seed(42)
data = np.random.normal(loc=0, scale=1, size=100)  # Generating random data

# Introduce outliers
data[5] = 8  # Adding an outlier
data[20] = -6  # Adding another outlier

# Function to detect outliers using IQR method
def detect_outliers_iqr(data):
```