```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
```

```
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Datasets/heart.csv')
```

```
df.shape
```

```
(918, 12)
```

```
df.columns.values
```

```
array(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol',
       'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina',
       'Oldpeak', 'ST_Slope', 'HeartDisease'],
      dtype=object)
```

```
df.dtypes
```

```
Age                 int64
Sex                object
ChestPainType      object
RestingBP           int64
Cholesterol         int64
FastingBS           int64
RestingECG         object
MaxHR               int64
ExerciseAngina     object
Oldpeak            float64
ST_Slope           object
HeartDisease        int64
dtype: object
```

```
df.head(5)
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

Data types of columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtype: int64, object(5), float64(1)
memory usage: 86.1+ KB
```

Table total changes are numbers so convert into numerics

```
df[['Sex','ChestPainType','RestingECG','ExerciseAngina','ST_Slope']].astype('category')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 0   Age             918 non-null    int64
 1   Sex             918 non-null    category
 2   ChestPainType   918 non-null    category
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    category
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    category
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    category
 11  HeartDisease    918 non-null    int64
dtypes: category(5), float64(1), int64(6)
memory usage: 47.1 KB
```

```
df.isnull().sum()
```

```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

```
df.dropna(inplace=True)
```

```
df.isnull().sum()
```

```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

some pandas method abbreviations contain in parenthesis the word axis/rows. These descriptions are too long to be used in full types a further explanation.

```
df.duplicated(keep=False).sum()
```

```
0
```

```
df[df.duplicated(keep=False)].sort_values(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope', 'HeartDisease'])
```

```
df.head(5)
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

DATA Visualisation

find check the dium values

```
df['Sex'].value_counts()
```

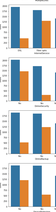
```
Sex
M    725
F    193
Name: count, dtype: int64
```

```
sns.countplot(df['Sex'],palette='Pastel1',kind='point',data=df)
```

```
<ipykrit.axisgrid.FacetGrid at 0x7f9f9f9f5f20>
```

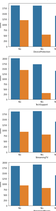
```
sns.countplot(df['Age'],palette='Pastel1',kind='point',data=df)
```

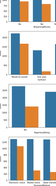
```
<ipykrit.axisgrid.FacetGrid at 0x7f9f9f9f5f20>
```

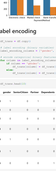


from this we observed that the people are having patient and chunks highly than Male is having

```
for i, j, k,l,m,n,o,p,q,r,s,t in df.drop(['Cholesterol','Oldpeak','RestingBP','MaxHR','Age','Sex','ChestPainType','RestingECG','FastingBS','ExerciseAngina','ST_Slope','HeartDisease'],axis=1).columns:
    sns.countplot(df[i],palette='Pastel1',kind='point',data=df)
    plt.show()
```







label encoding

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 0   Age             918 non-null    int64
 1   Sex             918 non-null    category
 2   ChestPainType   918 non-null    category
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    category
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    category
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    category
 11  HeartDisease    918 non-null    int64
dtypes: category(5), float64(1), int64(6)
memory usage: 47.1 KB
```

```
# label encoding
# pipe = Pipeline(steps=[('ohe', OneHotEncoder(sparse=False,handle_unknown='ignore')),
# ('scaler', StandardScaler())])
# pipe.fit(df[['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']])
from sklearn.preprocessing import LabelEncoder
for i in df[['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']]:
    df[i] = LabelEncoder().fit_transform(df[i])
    print('{} : {}'.format(i, LabelEncoder().fit(df[i]).classes_))
```

```
Sex : [0 1]
ChestPainType : [0 1 2 3]
RestingECG : [0 1 2]
ExerciseAngina : [0 1]
ST_Slope : [0 1 2]
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	1	1	140	289	0	1	172	0	0.0	2	0
1	49	0	2	160	180	0	1	156	0	1.0	1	1
2	37	1	1	130	283	0	2	98	0	0.0	2	0
3	48	0	0	138	214	0	1	108	1	1.5	1	1
4	54	1	2	150	195	0	1	122	0	0.0	2	0

here we converted the gin a to value this numerical and retraining will be converted with the one hot encoding.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 0   Age             918 non-null    int64
 1   Sex             918 non-null    int64
 2   ChestPainType   918 non-null    int64
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    int64
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    int64
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    int64
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(11)
memory usage: 86.1 KB
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
df[['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope', 'HeartDisease']]=sc.fit_transform(df[['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope', 'HeartDisease']])
df.head(5)
```

feature scaling

```
# from sklearn.preprocessing import StandardScaler
# sc=StandardScaler()
# df[['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope', 'HeartDisease']]=sc.fit_transform(df[['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope', 'HeartDisease']])
# df.head(5)
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	-1.43251	0.515689	-0.617936	0.410909	0.825070	0.296182	0.187563	1.382928	-0.823556	0.148182	0.981865	-0.550068
1	-0.478516	-1.939706	0.226996	1.491752	-0.171961	-0.545314	-0.544732	-0.754157	-0.823556	0.784932	0.570201	0.818965
2	-1.751359	0.515689	-0.617936	-0.129513	0.770188	-0.545314	1.619859	-1.525138	-0.823556	-0.149230	0.981865	-0.550068
3	-0.584444	-1.939706	-2.462867	0.302826	0.158067	0.296182	-0.544732	-1.132156	1.214246	0.529896	-2.274682	0.818965
4	-0.478516	0.515689	1.071964	0.951332	-0.677051	-0.545314	-0.544732	-0.581981	-0.823556	-0.062916	0.570201	-0.550068

train and testing

```
df[df.columns.values[:-1]].sample(5)
```

```
df[df.columns.values[-1]].sample(5)
```

```
918    0
918    1
918    0
918    1
918    1
Name: HeartDisease, dtype: int64
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df[df.columns.values[:-1]],df[df.columns.values[-1]],test_size=0.2)
```

```
print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
(734, 11) (734,)
(184, 11) (184,)
```

```
# logic
```

```
# from sklearn import
```

```
from sklearn.linear import LinearRegression
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df[df.columns.values[:-1]],df[df.columns.values[-1]],
                                                    random_state=0)
```

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(X_train,y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False,
                   fit_intercept=True, intercept_scaling=1,
                   l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001,
                   verbose=0, warm_start=False)
```

```
y_pred=lr.predict(X_test)
y_pred[:10]
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```