```
import  pandas as pd
import numpy as np
import matplotlib.pylab as plt
import seaborn as sns
from sklearn import linear_model
```

```
# Liner regression
```

```
#1. load Data
```

```
from sklearn.datasets import load_boston
booston_dataset = load_boston()
booston_dataset  =
pd.DataFrame(booston_dataset.data,columns=booston_dataset.feature_name
s)
```

```
C:\Users\Lenovo\AppData\Roaming\Python\Python310\site-packages\
sklearn\utils\deprecation.py:87: FutureWarning: Function load_boston
is deprecated; `load_boston` is deprecated in 1.0 and will be removed
in 1.2.

    The Boston housing prices dataset has an ethical problem. You can
refer to
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use
of this
    dataset unless the purpose of the code is to study and educate
about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::

        import pandas as pd
        import numpy as np

        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22,
header=None)
        data = np.hstack([raw_df.values[::2, :],
raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames
housing
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
```

```
        housing = fetch_california_housing()

    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)

    for the Ames housing dataset.
  warnings.warn(msg, category=FutureWarning)
```

```
#2. create Dataframe
#X,y= load_boston(return_X_y=True)
#X =pd.DataFrame(X)
#X.head()
```

```
C:\Users\Lenovo\AppData\Roaming\Python\Python310\site-packages\
sklearn\utils\deprecation.py:87: FutureWarning: Function load_boston
is deprecated; `load_boston` is deprecated in 1.0 and will be removed
in 1.2.

    The Boston housing prices dataset has an ethical problem. You can
refer to
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use
of this
    dataset unless the purpose of the code is to study and educate
about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::

        import pandas as pd
        import numpy as np

        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22,
header=None)
        data = np.hstack([raw_df.values[::2, :],
raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames
housing
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
        housing = fetch_california_housing()
```

```
    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)

    for the Ames housing dataset.
  warnings.warn(msg, category=FutureWarning)
```

```
         0     1     2    3      4      5     6       7    8      9
10  \
0  0.00632  18.0  2.31  0.0  0.538  6.575  65.2  4.0900  1.0  296.0
15.3
1  0.02731   0.0  7.07  0.0  0.469  6.421  78.9  4.9671  2.0  242.0
17.8
2  0.02729   0.0  7.07  0.0  0.469  7.185  61.1  4.9671  2.0  242.0
17.8
3  0.03237   0.0  2.18  0.0  0.458  6.998  45.8  6.0622  3.0  222.0
18.7
4  0.06905   0.0  2.18  0.0  0.458  7.147  54.2  6.0622  3.0  222.0
18.7

       11    12
0  396.90  4.98
1  396.90  9.14
2  392.83  4.03
3  394.63  2.94
4  396.90  5.33
```

```
booston_dataset.head()
```

```
      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  1.0  296.0

1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0

2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0

3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0

4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0


   PTRATIO       B  LSTAT
0     15.3  396.90   4.98
1     17.8  396.90   9.14
2     17.8  392.83   4.03
3     18.7  394.63   2.94
4     18.7  396.90   5.33
```

```
booston_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
dtypes: float64(13)
memory usage: 51.5 KB
```

```
#4. create X and Y
X =booston_dataset
```

```
kashti = sns.load_dataset('titanic')
```

```
kashti
```

```
     survived  pclass     sex   age  sibsp  parch     fare embarked  
class  \
0           0       3    male  22.0      1      0   7.2500        S  
Third   
1           1       1  female  38.0      1      0  71.2833        C  
First   
2           1       3  female  26.0      0      0   7.9250        S  
Third   
3           1       1  female  35.0      1      0  53.1000        S  
First   
4           0       3    male  35.0      0      0   8.0500        S  
Third   
..        ...     ...     ...   ...    ...    ...      ...      ...  
...   
886         0       2    male  27.0      0      0  13.0000        S  
Second  
887         1       1  female  19.0      0      0  30.0000        S  
First   
888         0       3  female   NaN      1      2  23.4500        S  
Third   
889         1       1    male  26.0      0      0  30.0000        C  
```

```
First
890          0       3    male  32.0      0      0   7.7500        Q
Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
..     ...         ...  ...          ...   ...    ...
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]
```

```
kashti.head()
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
2         1       3  female  26.0      0      0   7.9250        S
Third
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

```
# remove all null
#kashti.dropna(inplace=True)
#kashti

# remove null only from age col
kashti.dropna(subset= ['age'],inplace=True)
kashti
```

```
     survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
```

```
0           0       3    male  22.0      1      0   7.2500        S
Third
1           1       1  female  38.0      1      0  71.2833        C
First
2           1       3  female  26.0      0      0   7.9250        S
Third
3           1       1  female  35.0      1      0  53.1000        S
First
4           0       3    male  35.0      0      0   8.0500        S
Third
..        ...     ...     ...   ...    ...    ...      ...      ...
...
885         0       3  female  39.0      0      5  29.1250        Q
Third
886         0       2    male  27.0      0      0  13.0000        S
Second
887         1       1  female  19.0      0      0  30.0000        S
First
889         1       1    male  26.0      0      0  30.0000        C
First
890         0       3    male  32.0      0      0   7.7500        Q
Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
..     ...         ...  ...          ...   ...    ...
885  woman       False  NaN   Queenstown    no  False
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True

[714 rows x 15 columns]
```

```
# define X and y

X= kashti[['age']]
y= kashti[['fare']]
```

```
# check null value in data sey
X.isnull().sum()
y.isnull().sum()
```

```
fare    0
dtype: int64
```

```
X.shape
```

```
(714, 1)
```

```
y.shape
```

```
(714, 1)
```

```
kashti.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 714 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     714 non-null    int64
 1   pclass       714 non-null    int64
 2   sex          714 non-null    object
 3   age          714 non-null    float64
 4   sibsp        714 non-null    int64
 5   parch        714 non-null    int64
 6   fare         714 non-null    float64
 7   embarked     712 non-null    object
 8   class        714 non-null    category
 9   who          714 non-null    object
 10  adult_male   714 non-null    bool
 11  deck         184 non-null    category
 12  embark_town  712 non-null    object
 13  alive        714 non-null    object
 14  alone        714 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 70.2+ KB
```

```
#linear Regersion
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

```
X_train ,X_test,y_train ,y_test =
train_test_split(X,y ,test_size=0.2,random_state=0)
```

```
# train the model

model =LinearRegression()
model.fit(X_train,y_train)
```

```
LinearRegression()
```

```
# test the model
y_pred =model.predict(X_test)
```

```
# metrics to check the perfoermance of the model

from sklearn.metrics import mean_squared_error
```

```python
y_pred =model.predict(X_test)
mean_squared_error(y_test,y_pred)
```

```
2128.1892760608985
```

```python
from sklearn.metrics import
r2_score,mean_absolute_error,mean_absolute_error
import numpy as np
r2_score(y_test ,y_pred)
```

```
0.015800553474554446
```

```python
mae = mean_absolute_error(y_test ,y_pred)
mse = mean_squared_error(y_test ,y_pred)
rmse =np.sqrt(mse)
r2=r2_score(y_test ,y_pred)

print('Mean Absolute Error ',mae)
print('Mean Squared Erorr',mse)
print('Root Mean Squared Error',rmse)
print('R2 Score',r2 )
```

```
Mean Absolute Error  31.184262095057967
Mean Squared Erorr 2128.1892760608985
Root Mean Squared Error 46.13230187255887
R2 Score 0.015800553474554446
```

```python
mse = mean_squared_error(y_test ,y_pred)
mse
```

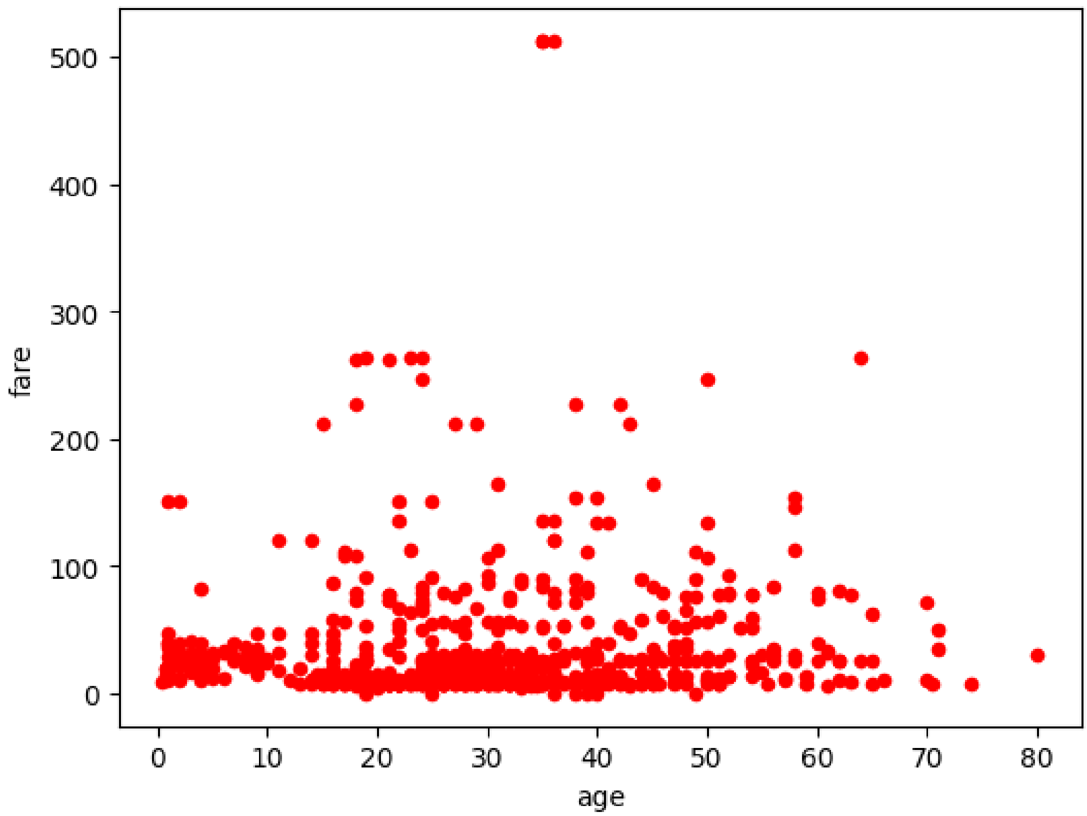
```
2128.1892760608985
```

```python
rmse =np.sqrt(mse)
rmse
```

```
46.13230187255887
```

```python
#plot
kashti.plot(kind='scatter', x='age',y='fare',color='red')
```

```
<AxesSubplot:xlabel='age', ylabel='fare'>
```



Mean Absolute Error 31.184262095057967

Mean Absolute Error (MAE): When we subtract the predicted values from the actual values, obtaining the errors, sum the absolute values of those errors and get their mean. This metric gives a notion of the overall error for each prediction of the model, the smaller (closer to 0) the better.

Mean Squared Erorr 2128.1892760608985

Mean Squared Error (MSE): It is similar to the MAE metric, but it squares the absolute values of the errors. Also, as with MAE, the smaller, or closer to 0, the better. The MSE value is squared so as to make large errors even larger. One thing to pay close attention to, it that it is usually a hard metric to interpret due to the size of its values and of the fact that they aren't in the same scale of the data.

Root Mean Squared Error 46.13230187255887

Root Mean Squared Error (RMSE): Tries to solve the interpretation problem raised with the MSE by getting the square root of its final value, so as to scale it back to the same units of the data. It is easier to interpret and good when we need to display or show the actual value of the data with the error. It shows how much the data may vary, so, if we have an RMSE of 4.35, our model can make an error either because it added 4.35 to the actual value, or needed 4.35 to get to the actual value. The closer to 0, the better as well.

R2 Score 0.015800553474554446

R-Squared (R^2 or the coefficient of determination) is a statistical measure in a regression model that determines the proportion of variance in the dependent variable that can be explained by the independent variable. In other words, r-squared shows how well the data fit the regression model (the goodness of fit).