22/7/25 Preprocessing using pandas and simple imputer.

AIM:

To load titanic dataset from csv, handle missing values using simple imputer, analyze key passenger features, filter passenger based on candidates, and prepare data for model training and testing.

Procedure / Algorithm:

Step 1: load titanic.csv into a dataframe

Step 2: Explore dataset shape, info and summary statistics.

Step 3: use simple Imputer to fill missing Age.

Step 4: Fill missing cabin with "unknown" and embaced with mode.

Step 5: visualize passenger class.

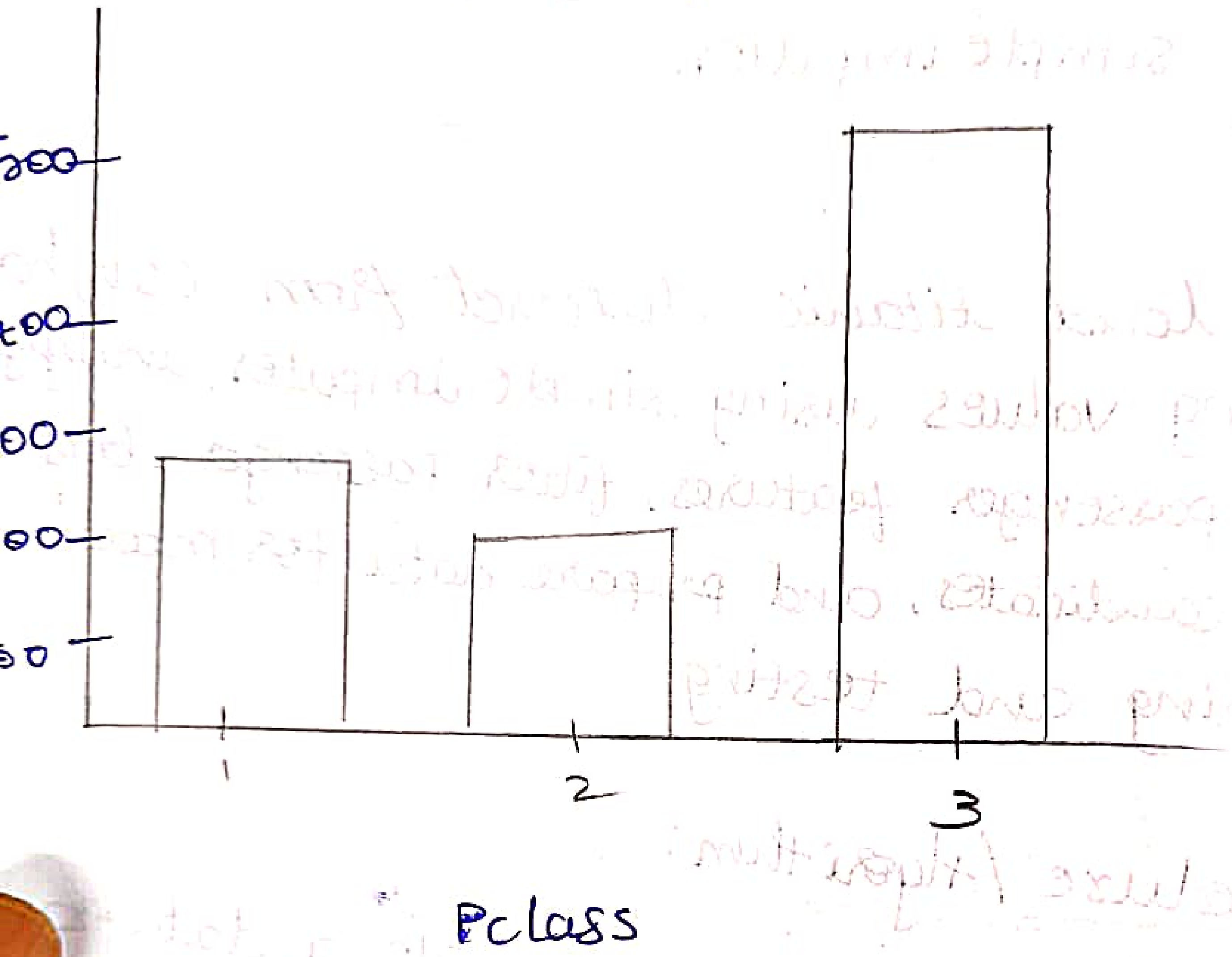
Step 6: Filter passangers by genders, survival, class, age fav embancation, family abroad, and survival status.

Step 7: Identity top oldest survivors and zeros - fave passangers

Step 8: Split training and testing sets.

output:

Passanger class distribution



PROGRAM:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from Sklearn.impute import Simple Imputer
from Sklearn.model_selection import train_
                                      test_split

df = sns.load_dataset ('titanic')
df['age'] = Simple Imputer (Strategy = 'mean')
            fit_transform (df[['age']])
df['deck'] = df['deck'].cate.add_categories
                                      ('unknown')
df['deck] = df['deck'].fillna('unknown')
df['embanced'] = df['embanced'].fillna
        (df['embanced'].mode()[0])
Sns.countplot (x = 'pclass', data=df) Plt.title
        ('Passanger class distribution')
  plt.show()
Print ("Females who survived:", df[(df.
   sex = 'female') & (df.survived ==1)].inde
                              .tolist()
Print(" 3rd class passangers under 18:",
   df[(df.pclass ==3) & (df.age<18)].
                         index.tolist()
```

Passangers who paid zero tax : 15 passangers

Training set size : 712

Testing set size = 179

```
Print (" (1st class passangers older than "
    df[(df. pclass ==1) & (df. age>40)].
                              index .to list())

Print (" 1st class passangers older than 40
    who survived: ", df [(df. Pclass ==1) & (df age>
    40) & (df-survived ==1) ].index. to list())
```

RESULT:

The Program successfully ind~~e~~entifies passangers with zero fax and efficiently splits the datasets into 80% training and 20% testing sets, ensuring reproducibility and readiness for machine learning tasks.

EX.NO:3 Model Planning and Building

Aim:

To describe the model planning and Building of the whole data set.

Code:

```
import pandas as pd
import matplotlib .pyplot as plt
import seaborn as sns
from sklearn. linear model import linear Regression
from sklearn metrices - selection import
                                train - test split
from sklearn metric .import mean _squared
from sklearn metric .import mean - squared
df = pd. read _cs ('advertising csv')
Print (df.head ())
Print (df.describe())
x = df ['TV', 'Radio', 'Newspaper']]
Y = df ['Sales']
# split data
```

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69..2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.7	69..3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

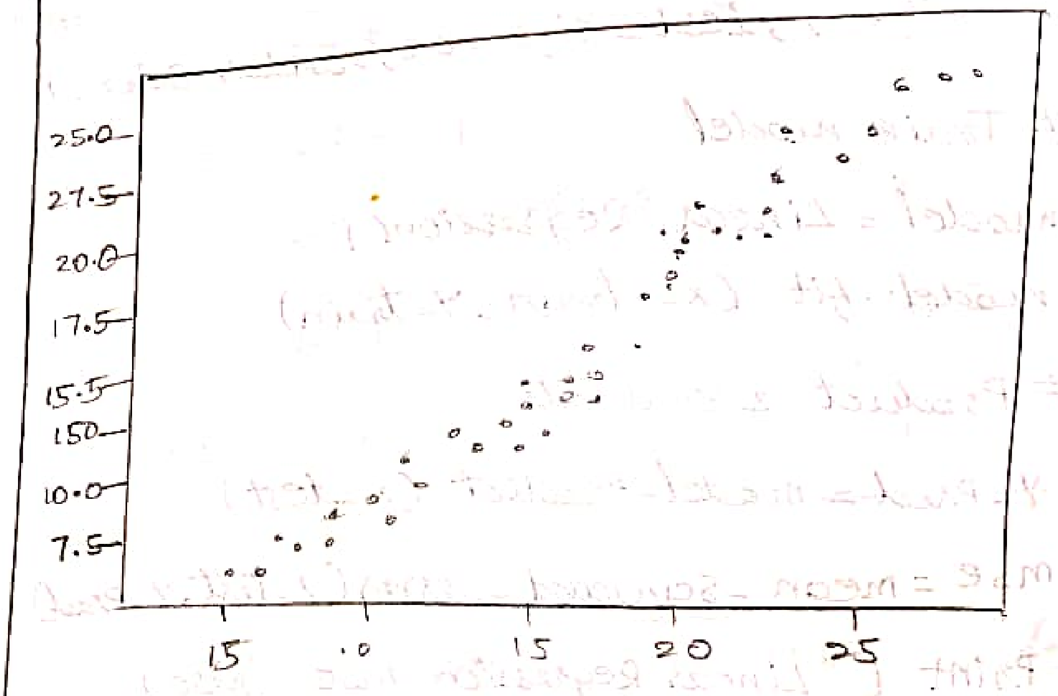
	TV	Radio	Newspaper	Sale
count	200.0000	2000.000	2000.0000	200.000
mean	147.0425	23.2640	30.55400	15.130500
Std	85.884	14.846	21.7786	5.2889
min	0.7000	0.000	0.3000	1.60000
25%	29.33730	9.975	12.78000	11.0000
50%	149.7500	22.900	25.78000	16.0000
max	296.4000	49.600	114.000	27.0000

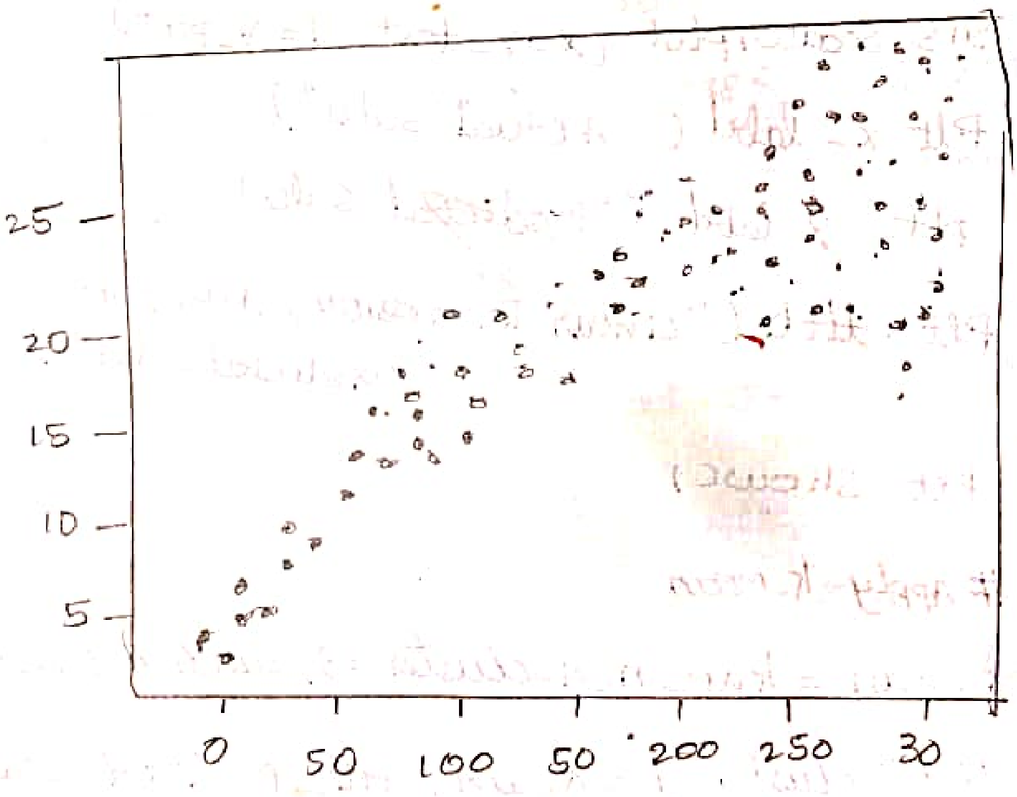
Linear Regression MSE : 4.522582562041291

```
x_train, x_test, y_train, y_test = train_test
                                        -split
        (x, y, test_ size = 0.2, random -sale = 0)
# Train model
 model = Linear Regression()
 model .fit (x- train, y-train)
# Product & Evaluate
 y_Pred = model-Prodict (x_test)
 mse = mean -squared - error (y_test, y_Pred)
  Print ("Linear Regression MSE ,mse)
  Plt · figure (fig size = (8,5))
   Sns. Scatterplot (x=y_test, y= y_plot)
   Plt. x- label ("Actual Sales")
    plt  y_ label ("predicted sales)
   Plt . title("Linear Regression: Actual vs
                          Predicted Sales")

    Plt show()
   # Apply k-mean
   k mean = kmean(n-cluster = 3, random-sale=0)
   df [" cluster "] = kman, fit_Predicted (scaled)
   plt . figure (figsize =(8,6))
```

Actual vs Predicted Sale





```
Sns. scatter plot (data=df , x='TV',
        y='sales')
Plt . title (k_mean clustering: Tv Budget
                                  vs Sales")
plt . show()
```

RESULT:

The Program has been executed successful.