```
from google.colab import files
upload=files.upload()
```

Choose Files BuyerRatio.csv

- **BuyerRatio.csv**(text/csv) - 57021 bytes, last modified: 4/1/2023 - 100% done

```
Saving BuyerRatio.csv to BuyerRatio.csv
```

```
import pandas as pd
import numpy as np
```

```
df=pd.read_csv("BuyerRatio.csv")
df.head()
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	wri s
0	female	group B	bachelor's degree	standard	none	72	72	
1	female	group C	some college	standard	completed	69	90	
			master's					

performing eda

```
df.describe()
```

	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

```
len(df)
```

```
1000
```

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

```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64 
 6   reading score                1000 non-null   int64 
 7   writing score                1000 non-null   int64 
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

```
df.value_counts()
```

```
gender  race/ethnicity  parental level of education  lunch         test preparation
course  math score  reading score  writing score
female  group A         associate's degree           free/reduced  none
37          57             56               1
male    group C         associate's degree           standard      completed
57          54             56               1
                                                     free/reduced  completed
60          51             56               1

65          67             65               1

73             68               1

..
female  group D         associate's degree           standard      none
71          71             74               1

74          81             83               1

76          74             73               1
```

```
77          77             73               1
male    group E         some high school            standard      none
94          88             78               1
Length: 1000, dtype: int64
```

```
df.gender
```

```
0      female
1      female
2      female
3        male
4        male
        ...
995    female
996      male
997    female
998    female
999    female
Name: gender, Length: 1000, dtype: object
```

```
f=df[df["gender"]=="female"]
```

```
m=df[df["gender"]=="male"]
```

```
df.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
995    False
996    False
997    False
998    False
999    False
Length: 1000, dtype: bool
```

```
##checking the howmany categires present in the column
```

```
df["gender"].unique()
```

```
array(['female', 'male'], dtype=object)
```

```
df['race/ethnicity'].unique()
```

```
      array(['group B', 'group C', 'group A', 'group D', 'group E'],
            dtype=object)
```

```
df["parental level of education"].unique()
```

```
      array(["bachelor's degree", 'some college', "master's degree",
             "associate's degree", 'high school', 'some high school'],
            dtype=object)
```

```
df["lunch"].unique()
```

```
      array(['standard', 'free/reduced'], dtype=object)
```

```
df["test preparation course"].unique()
```

```
      array(['none', 'completed'], dtype=object)
```

```
### separating the con and cat variable
```

```
con=[feature for  feature in df.columns if df[feature].dtype!="obj"]
```

```
con
```

```
      ['gender',
       'race/ethnicity',
       'parental level of education',
       'lunch',
       'test preparation course',
       'math score',
       'reading score',
       'writing score']
```

```
df.columns
```

```
      Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
             'test preparation course', 'math score', 'reading score',
             'writing score'],
            dtype='object')
```

```
cat_feat=df.loc[:,df.dtypes==np.object]
```

```
      <ipython-input-23-4e57153ddcd0>:1: DeprecationWarning: `np.object` is a deprecated alias
      Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/relea
        cat_feat=df.loc[:,df.dtypes==np.object]
```

```
con_feat=df.loc[:,df.dtypes==np.int]
```

```
<ipython-input-24-0f03346fccb7>:1: DeprecationWarning: `np.int` is a deprecated alias fo
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/relea
  con_feat=df.loc[:,df.dtypes==np.int]
```

```
df["total score"]=df["math score"]+df["reading score"]+df["writing score"]
```

```
df["Average"]=round(df["total score"]/3,2)
```

```
df.head()
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
			master's					

```
df[(df["reading score"]== 100) & (df["math score"] == 100)]
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
149	male	group E	associate's degree	free/reduced	completed	100	100	93
458	female	group E	bachelor's degree	standard	none	100	100	100
916	male	group E	bachelor's	standard	completed	100	100	100

```
df[df["math score"]==100]
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
149	male	group E	associate's degree	free/reduced	completed	100	100	93
451	female	group E	some college	standard	none	100	92	97

```
df[df["math score"]==100]["Average"].count()
```

```
7
```

623	male	group A	some ...	standard	completed	100	96	86

```
df[df["reading score"]==100]["Average"].count()
```

```
17
```

```
df[df["writing score"]==100]["Average"].count()
```

```
14
```

```
df[df["reading score"]<=20]["Average"].count()
```

```
1
```

```
df[df["writing score"]<=20]["Average"].count()
```

```
3
```

```
df[df["math score"]<=20]["Average"].count()
```

```
4
```

```
df[df["reading score"]<=20]["Average"].count()
```

```
1
```

```
df[df["writing score"]<=20]["Average"].count()
```

```
3
```

```
## from above we can say that the students are performed well in reading sccore and weak perf
```

```
df.nunique()
```

```
gender                         2
race/ethnicity                 5
parental level of education    6
lunch                          2
test preparation course        2
math score                    81
reading score                 72
writing score                 77
total score                  194
Average                      194
dtype: int64
```

performing EDa

```
pip install  AutoViz
```

```
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-pa
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.9/dist-pac
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.9/dist-packa
Requirement already satisfied: fastjsonschema in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: argon2-cffi-bindings in /usr/local/lib/python3.9/dist-
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr/
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: pycparser in /usr/local/lib/python3.9/dist-packages (f
Building wheels for collected packages: emoji
  Building wheel for emoji (setup.py) ... done
  Created wheel for emoji: filename=emoji-2.2.0-py3-none-any.whl size=234926 sha256=e
  Stored in directory: /root/.cache/pip/wheels/9a/b8/0f/f580817231cbf59f6ade9fd132ff6
Successfully built emoji
Installing collected packages: qtpy, jedi, emoji, pyamg, qtconsole, hvplot, jupyter, A
Successfully installed AutoViz-0.1.601 emoji-2.2.0 hvplot-0.8.3 jedi-0.18.2 jupyter-1
```

```
from autoviz.AutoViz_Class import AutoViz_Class

AV = AutoViz_Class()
```

```
Imported v0.1.601. After importing, execute '%matplotlib inline' to display charts in Ju
    AV = AutoViz_Class()
    dfte = AV.AutoViz(filename, sep=',', depVar='', dfte=None, header=0, verbose=1, lowe
                chart_format='svg',max_rows_analyzed=150000,max_cols_analyzed=30, save_pl
Update: verbose=0 displays charts in your local Jupyter notebook.
        verbose=1 additionally provides EDA data cleaning suggestions. It also displays
        verbose=2 does not display charts but saves them in AutoViz_Plots folder in loca
        chart_format='bokeh' displays charts in your local Jupyter notebook.
        chart_format='server' displays charts in your browser: one tab for each chart ty
        chart_format='html' silently saves interactive HTML files in your local machine
```

```
filename = "BuyerRatio.csv"
sep = ","
dft = AV.AutoViz(
    filename,
    sep=",",
    depVar="",
    dfte=None,
    header=0,
    verbose=0,
    lowess=False,
    chart_format="svg",
    max_rows_analyzed=150000,
    max_cols_analyzed=30,
    save_plot_dir=None
)
```

```
Shape of your Data Set loaded: (1000, 8)
################################################################################################
######################## C L A S S I F Y I N G  V A R I A B L E S  ##############################
################################################################################################
Classifying variables in data set...
    Number of Numeric Columns =  0
    Number of Integer-Categorical Columns =  3
    Number of String-Categorical Columns =  2
    Number of Factor-Categorical Columns =  0
    Number of String-Boolean Columns =  3
    Number of Numeric-Boolean Columns =  0
    Number of Discrete String Columns =  0
    Number of NLP String Columns =  0
    Number of Date Time Columns =  0
    Number of ID Columns =  0
    Number of Columns to Delete =  0
    8 Predictors classified...
        No variables removed since no ID or low-information variables found in data set
Number of All Scatter Plots = 6
All Plots done
Time to run AutoViz = 3 seconds

 ###################### AUTO VISUALIZATION Completed ########################
```

```
pip install SweetViz
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting SweetViz
  Downloading sweetviz-2.1.4-py3-none-any.whl (15.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.1/15.1 MB 78.1 MB/s eta 0:00:00
Requirement already satisfied: tqdm>=4.43.0 in /usr/local/lib/python3.9/dist-packages (f
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.9/dist-packages (f
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: matplotlib>=3.1.3 in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: jinja2>=2.11.1 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: importlib-resources>=1.2.0 in /usr/local/lib/python3.9/di
Requirement already satisfied: pandas!=1.0.0,!=1.0.1,!=1.0.2,>=0.25.3 in /usr/local/lib/
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.9/dist-packages (fr
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-packages (f
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.9/dist-pac
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (f
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from
Installing collected packages: SweetViz
Successfully installed SweetViz-2.1.4
```

```
import sweetviz

df= pd.read_csv("BuyerRatio.csv")

my_report = sweetviz.analyze([df, "Train"],target_feat='reading score')
```

Done! Use 'show' commands to display/save. [100%] 00:01 -> (00:00 left)

```
Y=df["math score"]

Y
```

```
0      72
1      69
2      90
3      47
4      76
       ..
995    88
996    62
997    59
998    68
999    77
Name: math score, Length: 1000, dtype: int64
```

```
### create column  transformer( to convert data into one hot and label) at a time

from sklearn.preprocessing import OneHotEncoder, StandardScaler

SS = StandardScaler()

for col in con_feat:
  SS=StandardScaler()
  con_feat[col]=SS.fit_transform(con_feat[[col]])

con_feat
```

	math score	reading score	writing score
0	0.390024	0.193999	0.391492
1	0.192076	1.427476	1.313269
2	1.577711	1.770109	1.642475
3	-1.259543	-0.833899	-1.583744
4	0.653954	0.605158	0.457333
...	...	...	...
995	1.445746	2.044215	1.774157
996	-0.269803	-0.970952	-0.859491
997	-0.467751	0.125472	-0.201079
998	0.126093	0.605158	0.589015

```
from sklearn.preprocessing import LabelEncoder
LE = LabelEncoder()
for i in range(0,5):
  cat_feat.iloc[:,i]=LE.fit_transform(cat_feat.iloc[:,i])
```

```
cat_feat
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course
0	0	1	1	1	1
1	0	2	4	1	0
2	0	1	3	1	1
3	1	0	0	0	1
4	1	2	4	1	1
...	...	...	...	...	...
995	0	4	3	1	0
996	1	2	2	0	1
997	0	2	2	0	0
998	0	3	4	1	0
999	0	3	4	0	1

1000 rows × 5 columns

```
df1=pd.concat([cat_feat,con_feat],axis=1)
```

```
df1
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	0	1	1	1	1	0.390024	0.193999	0.39149
1	0	2	4	1	0	0.192076	1.427476	1.31326
2	0	1	3	1	1	1.577711	1.770109	1.64247
3	1	0	0	0	1	-1.259543	-0.833899	-1.58374
4	1	2	4	1	1	0.653954	0.605158	0.45733
...	...	...	...	...	...	...	...	...
995	0	4	3	1	0	1.445746	2.044215	1.77415
996	1	2	2	0	1	-0.269803	-0.970952	-0.85949
997	0	2	2	0	0	-0.467751	0.125472	-0.20107
998	0	3	4	1	0	0.126093	0.605158	0.58901
999	0	3	4	0	1	0.719937	1.153370	1.18158

```
X=df1.drop(columns=["math score"],axis=1)
```

```
X
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	reading score	writing score

```
Y=df1["math score"]


from sklearn.model_selection import train_test_split
```

2	0	1	3	1	1	1.770109	1.642470

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=54)
```

4	1	2	4	1	1	0.605158	0.457333

```
X_train
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	reading score	writing score
221	1	1	0	1	1	1.084843	0.325651
514	0	1	3	0	0	1.907162	1.708316
917	0	2	2	1	1	-0.559793	-1.057015
258	0	1	4	1	1	0.399578	0.654857
517	0	4	4	1	0	0.331052	0.325651
...	...	...	...	...	...	...	...
23	0	2	5	1	1	0.262525	0.325651
783	0	2	0	1	0	-0.696846	-0.398603
898	1	3	0	1	0	-1.039479	-0.332761
325	0	2	4	1	1	1.427476	1.708316
111	1	2	2	1	1	-0.970952	-1.254538

```
Y_train
```

```
221    1.379763
514    0.719937
917   -1.457491
258    0.258058
517   -0.005872
         ...
23     0.192076
783   -0.929630
898    0.060110
325    1.049850
111   -0.269803
Name: math score, Length: 800, dtype: float64
```

```
##model evaluation
```

```
pip install catboost
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting catboost
  Downloading catboost-1.1.1-cp39-none-manylinux1_x86_64.whl (76.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.6/76.6 MB 9.3 MB/s eta 0:00:00
Requirement already satisfied: matplotlib in /usr/local/lib/python3.9/dist-packages (fro
Requirement already satisfied: plotly in /usr/local/lib/python3.9/dist-packages (from ca
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (from cat
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: graphviz in /usr/local/lib/python3.9/dist-packages (from
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: six in /usr/local/lib/python3.9/dist-packages (from catbo
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (f
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-p
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-packages (f
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: importlib-resources>=3.2.0 in /usr/local/lib/python3.9/di
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.9/dist-packages (fr
Installing collected packages: catboost
Successfully installed catboost-1.1.1
```

```
from sklearn.metrics import mean_squared_error,r2_score,mean_absolute_error
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression,Ridge,Lasso
from xgboost import XGBRegressor
from sklearn.model_selection import RandomizedSearchCV
from catboost import CatBoostRegressor
import warnings
```

KNN

```
kNN=KNeighborsRegressor(n_neighbors=4)
kNN.fit(X_train,Y_train)
Y_pred_train=kNN.predict(X_train)
Y_pred_test=kNN.predict(X_test)
```

```
print("knn for training set")
print("Root mean squared error",mean_squared_error(Y_train,Y_pred_train))
print("Root mean absolute error",mean_absolute_error(Y_train,Y_pred_train))
print("r2_score",r2_score(Y_train,Y_pred_train))
print(" ..................................")
print("knn performance for test set")
print("Root mean squared error",mean_squared_error(Y_test,Y_pred_test))
print("Root mean absolute error",mean_absolute_error(Y_test,Y_pred_test))
print("r2_score",r2_score(Y_test,Y_pred_test))
```

```
knn for training set
Root mean squared error 0.1381368054725841
Root mean absolute error 0.2998293359863729
r2_score 0.864552873502159
 ..................................
knn performance for test set
Root mean squared error 0.20165847763271327
Root mean absolute error 0.36191489617170874
r2_score 0.7780439958669642
```

DEcision tree regressor

```
#DT=DecisionTreeRegressor(criterion="squared_error",max_depth=None,max_features=7,min_samples
DT=DecisionTreeRegressor()
DT.fit(X_train,Y_train)
Y_pred_train=DT.predict(X_train)
Y_pred_test=DT.predict(X_test)
```

```
print("DT for training set")
print("Root mean squared error",mean_squared_error(Y_train,Y_pred_train))
print("Root mean absolute error",mean_absolute_error(Y_train,Y_pred_train))
print("r2_score",r2_score(Y_train,Y_pred_train))
print(" ..................................")
print("DT performance for test set")
print("Root mean squared error",mean_squared_error(Y_test,Y_pred_test))
print("Root mean absolute error",mean_absolute_error(Y_test,Y_pred_test))
print("r2_score",r2_score(Y_test,Y_pred_test))
```

```
DT for training set
Root mean squared error 0.0008843476619974606
Root mean absolute error 0.0025568281178949344
r2_score 0.9991328715816697
 ..................................
DT performance for test set
Root mean squared error 0.2386160466950194
Root mean absolute error 0.3728020352543582
r2_score 0.7373665373845071
```

Random forest

```
RM=RandomForestRegressor()
RM.fit(X_train,Y_train)
Y_pred_train=RM.predict(X_train)
Y_pred_test=RM.predict(X_test)
```

```
print("RM for training set")
print("Root mean squared error",mean_squared_error(Y_train,Y_pred_train))
print("Root mean absolute error",mean_absolute_error(Y_train,Y_pred_train))
print("r2_score",r2_score(Y_train,Y_pred_train))
print(" ..................................")
print("RM performance for test set")
print("Root mean squared error",mean_squared_error(Y_test,Y_pred_test))
print("Root mean absolute error",mean_absolute_error(Y_test,Y_pred_test))
print("r2_score",r2_score(Y_test,Y_pred_test))
```

```
RM for training set
Root mean squared error 0.023671706928256025
Root mean absolute error 0.12153039804845686
r2_score 0.9767892078308728
 ..................................
RM performance for test set
Root mean squared error 0.13780252854141717
Root mean absolute error 0.2987680363027821
r2_score 0.8483272364567337
```

ADA Boost

```
AB=AdaBoostRegressor()
AB.fit(X_train,Y_train)
Y_pred_train=AB.predict(X_train)
Y_pred_test=AB.predict(X_test)
```

```
print("AB for training set")
print("Root mean squared error",mean_squared_error(Y_train,Y_pred_train))
print("Root mean absolute error",mean_absolute_error(Y_train,Y_pred_train))
print("r2_score",r2_score(Y_train,Y_pred_train))
print(" ..................................")
print("AB performance for test set")
print("Root mean squared error",mean_squared_error(Y_test,Y_pred_test))
print("Root mean absolute error",mean_absolute_error(Y_test,Y_pred_test))
print("r2_score",r2_score(Y_test,Y_pred_test))
```

```
AB for training set
Root mean squared error 0.14316628537537035
```

```
Root mean absolute error 0.3074417420378279
r2_score 0.8596213232300902
 ..................................
AB performance for test set
Root mean squared error 0.15219305152411455
Root mean absolute error 0.32250393150955875
r2_score 0.8324882644674599
```

SVMt

```
SV=SVR()
SV.fit(X_train,Y_train)
Y_pred_train=SV.predict(X_train)
Y_Pred_test=SV.predict(X_test)
```

```
print("SV for training set")
print("Root mean squared error",mean_squared_error(Y_train,Y_pred_train))
print("Root mean absolute error",mean_absolute_error(Y_train,Y_pred_train))
print("r2_score",r2_score(Y_train,Y_pred_train))
print(" ..................................")
print("SV performance for test set")
print("Root mean squared error",mean_squared_error(Y_test,Y_pred_test))
print("Root mean absolute error",mean_absolute_error(Y_test,Y_pred_test))
print("r2_score",r2_score(Y_test,Y_pred_test))
```

```
SV for training set
Root mean squared error 0.12899127319608708
Root mean absolute error 0.28103457818806166
r2_score 0.8735203319786086
 ..................................
SV performance for test set
Root mean squared error 0.15219305152411455
Root mean absolute error 0.32250393150955875
r2_score 0.8324882644674599
```

LINEAR REGRESSIONt

```
LR=LinearRegression()
LR.fit(X_train,Y_train)
Y_pred_train=LR.predict(X_train)
Y_pred_test=LR.predict(X_test)
```

```
print("LR for training set")
print("Root mean squared error",mean_squared_error(Y_train,Y_pred_train))
print("Root mean absolute error",mean_absolute_error(Y_train,Y_pred_train))
print("r2_score",r2_score(Y_train,Y_pred_train))
```

```
print(" ..................................")
print("LR performance for test set")
print("Root mean squared error",mean_squared_error(Y_test,Y_pred_test))
print("Root mean absolute error",mean_absolute_error(Y_test,Y_pred_test))
print("r2_score",r2_score(Y_test,Y_pred_test))
```

```
LR for training set
Root mean squared error 0.13581941719802942
Root mean absolute error 0.29231722261546683
r2_score 0.8668251396204785
 ..................................
LR performance for test set
Root mean squared error 0.12287776372604375
Root mean absolute error 0.2893423476675329
r2_score 0.8647542233106116
```

XGBOOST

```
XG= XGBRegressor()
XG.fit(X_train,Y_train)
Y_pred_train=XG.predict(X_train)
Y_pred_test=XG.predict(X_test)
```

```
print("XG for training set")
print("Root mean squared error",mean_squared_error(Y_train,Y_pred_train))
print("Root mean absolute error",mean_absolute_error(Y_train,Y_pred_train))
print("r2_score",r2_score(Y_train,Y_pred_train))
print(" ..................................")
print("XG performance for test set")
print("Root mean squared error",mean_squared_error(Y_test,Y_pred_test))
print("Root mean absolute error",mean_absolute_error(Y_test,Y_pred_test))
print("r2_score",r2_score(Y_test,Y_pred_test))
```

```
XG for training set
Root mean squared error 0.004996752065032947
Root mean absolute error 0.04520203049853835
r2_score 0.9951005403178721
 ..................................
XG performance for test set
Root mean squared error 0.15840041083245118
Root mean absolute error 0.31351193695509044
r2_score 0.8256561159534472
```

catboost)

```
cat=CatBoostRegressor()
cat.fit(X_train,Y_train)
```

```
Y_pred_train=cat.predict(X_train)
Y_pres_test=cat.predict(X_test)
```

```
995:    learn: 0.2096091        total: 1.35s    remaining: 5.44ms
996:    learn: 0.2096049        total: 1.35s    remaining: 4.08ms
997:    learn: 0.2095429        total: 1.36s    remaining: 2.72ms
998:    learn: 0.2094952        total: 1.36s    remaining: 1.36ms
999:    learn: 0.2094451        total: 1.36s    remaining: 0us
```

```
print("cat for training set")
print("Root mean squared error",mean_squared_error(Y_train,Y_pred_train))
print("Root mean absolute error",mean_absolute_error(Y_train,Y_pred_train))
print("r2_score",r2_score(Y_train,Y_pred_train))
print(" ..................................")
print("cat performance for test set")
print("Root mean squared error",mean_squared_error(Y_test,Y_pred_test))
print("Root mean absolute error",mean_absolute_error(Y_test,Y_pred_test))
print("r2_score",r2_score(Y_test,Y_pred_test))
```

```
cat for training set
Root mean squared error 0.04386725338779256
Root mean absolute error 0.16299119055102843
r2_score 0.9569868913762566
 ..................................
cat performance for test set
Root mean squared error 0.15840041083245118
Root mean absolute error 0.31351193695509044
r2_score 0.8256561159534472
```

ridge regressor

```
from sklearn.model_selection import GridSearchCV
```

```
ridge=Ridge()
parameters={"alpha":[1,2,3,55,66]}
ridge_regressor=GridSearchCV(ridge,parameters,scoring="neg_mean_squared_error",cv=5)
ridge_regressor.fit(X_train,Y_train)
Y_pred_train=ridge_regressor.predict(X_train)
Y_pred_test=ridge_regressor.predict(X_test)
```

```
print("ridge for training set")
print("Root mean squared error",mean_squared_error(Y_train,Y_pred_train))
print("Root mean absolute error",mean_absolute_error(Y_train,Y_pred_train))
print("r2_score",r2_score(Y_train,Y_pred_train))
print(" ..................................")
print("ridge performance for test set")
print("Root mean squared error",mean_squared_error(Y_test,Y_pred_test))
print("Root mean absolute error",mean_absolute_error(Y_test,Y_pred_test))
print("r2_score",r2_score(Y_test,Y_pred_test))
```

```
ridge for training set
Root mean squared error 0.13583086470246167
Root mean absolute error 0.2922860623799845
r2_score 0.8668139150118332
 ..................................
ridge performance for test set
Root mean squared error 0.12292636026709264
Root mean absolute error 0.2893025977443636
r2_score 0.8647007353829403
```

```
pip install prettytable
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: prettytable in /usr/local/lib/python3.9/dist-packages (0
```

```
from prettytable import PrettyTable

# Specify the Column Names while initializing the Table
myTable = PrettyTable(["classifier", "train_mse", "test_mse", "train_mae","test_mae","train_r

# Add rows
myTable.add_row(["knn"," 0.13", "0.20", "0.29","0.36","0.86","0.77"])
myTable.add_row([" Decision tree", "0.0", "0.23", "0.00","0.37","0.99","0.73"])
myTable.add_row([" Randomforest", "0.02", "0.13", "0.12","0.29","0.97","0.84"])
myTable.add_row(["AdaBoost", "0.14", "0.15", "0.30","0.32","0.85","0.83"])
myTable.add_row(["SVM","0.12", "0.15", "0.28","0.32","0.87","0.83"])
myTable.add_row(["LinearRegression", "0.13", "0.12", "0.29","0.28","0.866","0.864"])
myTable.add_row(["XGB", "0.00", "0.15", "0.04","0.31","0.99","0.82"])
myTable.add_row(["catboost", "0.04", "0.15", "0.16","0.31","0.95","0.82"])

print(myTable)
```

```
+------------------+-----------+----------+-----------+----------+----------+---------+
|    classifier    | train_mse | test_mse | train_mae | test_mae | train_r2 | test_r2 |
+------------------+-----------+----------+-----------+----------+----------+---------+
|       knn        |    0.13   |   0.20   |    0.29   |   0.36   |   0.86   |   0.77  |
|   Decision tree  |    0.0    |   0.23   |    0.00   |   0.37   |   0.99   |   0.73  |
|   Randomforest   |    0.02   |   0.13   |    0.12   |   0.29   |   0.97   |   0.84  |
|     AdaBoost     |    0.14   |   0.15   |    0.30   |   0.32   |   0.85   |   0.83  |
|       SVM        |    0.12   |   0.15   |    0.28   |   0.32   |   0.87   |   0.83  |
| LinearRegression |    0.13   |   0.12   |    0.29   |   0.28   |  0.866   |  0.864  |
|       XGB        |    0.00   |   0.15   |    0.04   |   0.31   |   0.99   |   0.82  |
|     catboost     |    0.04   |   0.15   |    0.16   |   0.31   |   0.95   |   0.82  |
+------------------+-----------+----------+-----------+----------+----------+---------+
```

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
###scatter plot
df1.plot.scatter(x="gender", y="math score",title="Simple linear Regression",color="green")
```

```
<Axes: title={'center': 'Simple linear Regression'}, xlabel='gender', ylabel='math
score'>
```

```
df1.hist(None)
```

```
array([[<Axes: title={'center': 'gender'}>,
        <Axes: title={'center': 'race/ethnicity'}>,
        <Axes: title={'center': 'parental level of education'}>],
       [<Axes: title={'center': 'lunch'}>,
        <Axes: title={'center': 'test preparation course'}>,
        <Axes: title={'center': 'math score'}>],
       [<Axes: title={'center': 'reading score'}>,
        <Axes: title={'center': 'writing score'}>, <Axes: >]],
      dtype=object)
```

choosing the best mode *linear regression

```
LR=LinearRegression()
LR.fit(X_train,Y_train)
Y_Pred=LR.predict(X_test)
score=r2_score(Y_test,Y_pred_test)*100
print("the accuracy is",score)
```

```
the accuracy is 86.47007353829403
```

scatter plott

```
plt.scatter(Y_test,Y_pred_test)
plt.xlabel("actual")
plt.ylabel("predicted")
```

```
Text(0, 0.5, 'predicted')
```

difference betweeen actual and predicted value

```
pred_df=pd.DataFrame({"actual value":Y_test,"predicted value":Y_pred_test,"difference":Y_test
```

```
pred_df
```

	actual value	predicted value	difference
185	-0.071855	-0.020216	-0.051639
658	-1.523474	-1.068613	-0.454861
744	-0.731682	-0.868888	0.137206
970	1.511729	1.749528	-0.237799
129	-0.995612	-1.534605	0.538993
...	...	...	...
895	-2.249283	-2.351763	0.102480
276	1.247798	1.077687	0.170111
403	1.445746	1.554506	-0.108760
361	1.247798	1.081474	0.166324
793	1.511729	1.220500	0.291228

200 rows × 3 columns

```
pred_df.describe()
```

	actual value	predicted value	difference
count	200.000000	200.000000	200.000000
mean	0.098050	0.109436	-0.011385
std	0.955572	0.884513	0.351303
min	-2.579196	-2.658699	-0.856132
25%	-0.467751	-0.427747	-0.261135
50%	0.126093	0.125992	-0.003630
75%	0.736432	0.752350	0.234843
max	2.171555	1.865471	0.838479

0s completed at 6:20 PM