```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import catboost
from sklearn.preprocessing import LabelEncoder
from sklearn import metrics

import warnings
warnings.filterwarnings('ignore')
```

```
Sales = pd.read_csv('/content/Task  Dataset.csv')
Sales.head()
```

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

```
Sales = Sales.loc[:, ~Sales.columns.str.contains('^Unnamed')]
```

```
Sales.head()
```

	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	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

```
Sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

```
Sales.describe()
```

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	14.022500
std	85.854236	14.846809	21.778621	5.217457
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	10.375000
50%	149.750000	22.900000	25.750000	12.900000
75%	218.825000	36.525000	45.100000	17.400000
max	296.400000	49.600000	114.000000	27.000000

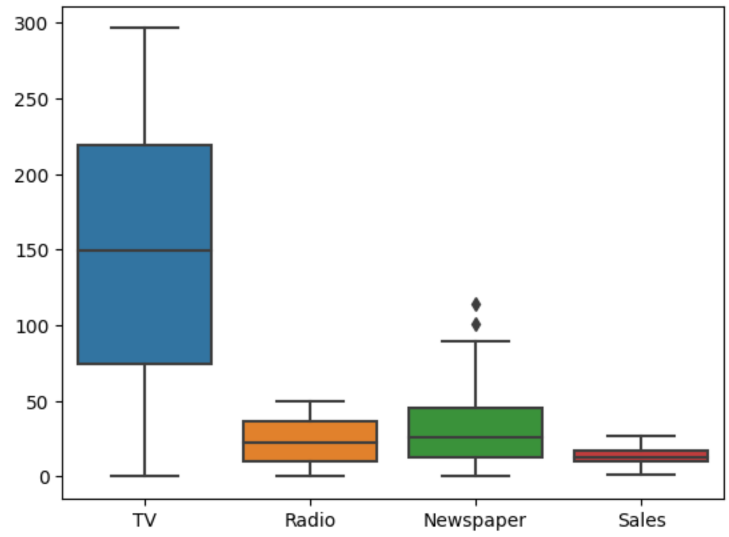
```
Sales.isnull().sum()
```

```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

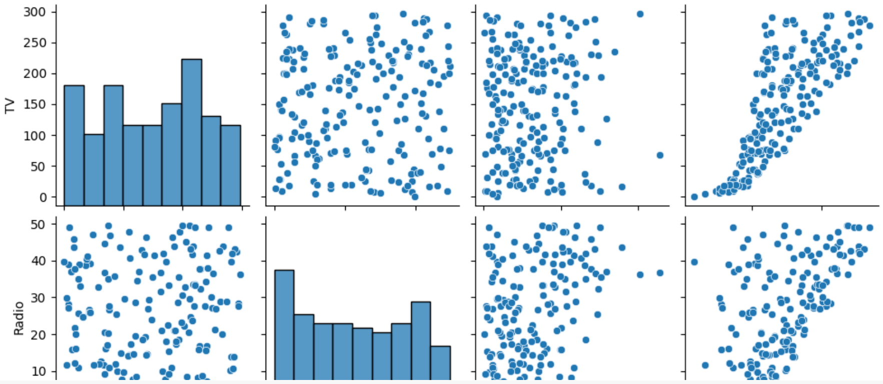
```
Sales.duplicated().sum()
```

```
0
```

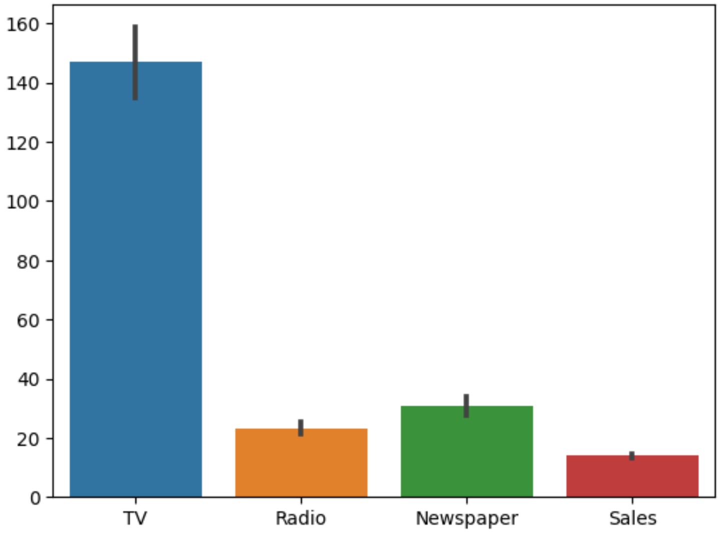
```
sns.boxplot(Sales);
```



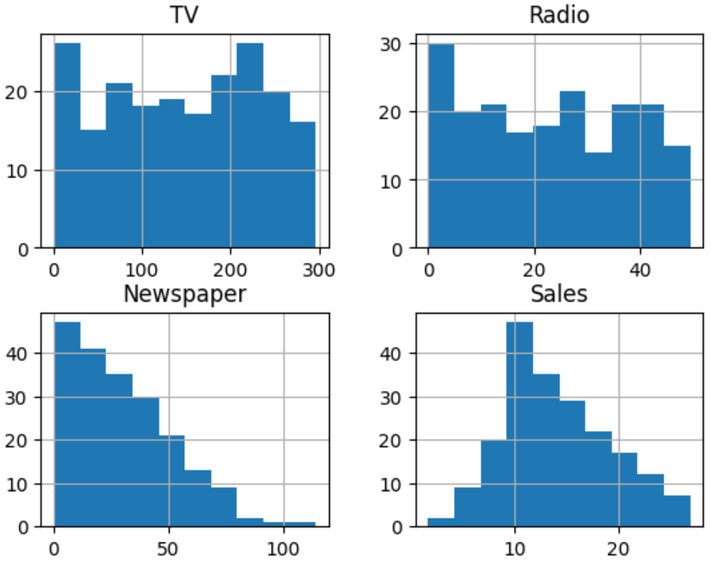
```
sns.pairplot(Sales);
```



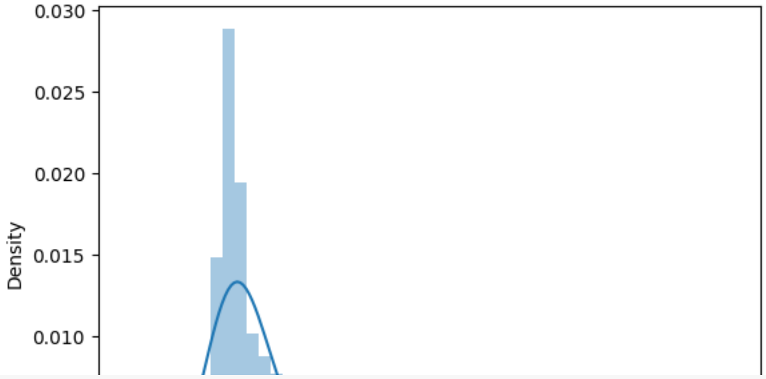
```
sns.barplot(Sales);
```



```
Sales.hist();
```



```
sns.distplot(Sales);
```

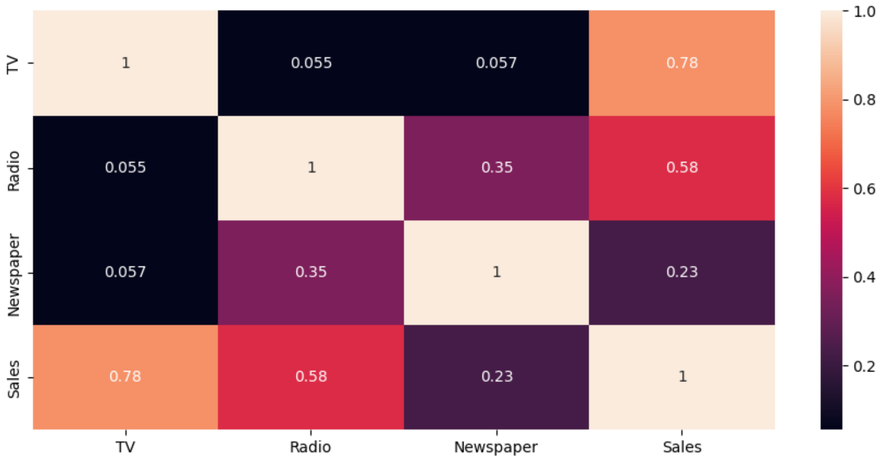


```
corr = Sales.corr()
corr
```

	TV	Radio	Newspaper	Sales
TV	1.000000	0.054809	0.056648	0.782224
Radio	0.054809	1.000000	0.354104	0.576223
Newspaper	0.056648	0.354104	1.000000	0.228299
Sales	0.782224	0.576223	0.228299	1.000000

```
fig, ax = plt.subplots(figsize = (11,5))
sns.heatmap(corr, annot=True, ax=ax)
```

```
<Axes: >
```



```
X = Sales.drop(['Sales'], axis=1)
y = Sales['Sales']
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.75, random_state=3332)
```

```
from catboost import CatBoostRegressor

CBR = CatBoostRegressor()
```

```
CBR.fit(X_train, y_train)
```

```
947:	learn: 0.1086087	total: 576ms	remaining: 31.6ms
948:	learn: 0.1083905	total: 577ms	remaining: 31ms
949:	learn: 0.1082987	total: 577ms	remaining: 30.4ms
950:	learn: 0.1080520	total: 578ms	remaining: 29.8ms
951:	learn: 0.1079078	total: 578ms	remaining: 29.2ms
952:	learn: 0.1076959	total: 579ms	remaining: 28.6ms
953:	learn: 0.1074686	total: 580ms	remaining: 27.9ms
954:	learn: 0.1072773	total: 580ms	remaining: 27.3ms
955:	learn: 0.1072428	total: 581ms	remaining: 26.7ms
956:	learn: 0.1071343	total: 581ms	remaining: 26.1ms
957:	learn: 0.1069682	total: 582ms	remaining: 25.5ms
958:	learn: 0.1067875	total: 583ms	remaining: 24.9ms
959:	learn: 0.1065359	total: 583ms	remaining: 24.3ms
960:	learn: 0.1064972	total: 584ms	remaining: 23.7ms
961:	learn: 0.1062934	total: 584ms	remaining: 23.1ms
962:	learn: 0.1060696	total: 585ms	remaining: 22.5ms
963:	learn: 0.1058576	total: 586ms	remaining: 21.9ms
964:	learn: 0.1055429	total: 586ms	remaining: 21.3ms
965:	learn: 0.1053139	total: 587ms	remaining: 20.6ms
966:	learn: 0.1050585	total: 587ms	remaining: 20ms
967:	learn: 0.1047257	total: 588ms	remaining: 19.4ms
968:	learn: 0.1045508	total: 588ms	remaining: 18.8ms
969:	learn: 0.1042904	total: 589ms	remaining: 18.2ms
970:	learn: 0.1042041	total: 590ms	remaining: 17.6ms
971:	learn: 0.1041738	total: 590ms	remaining: 17ms
972:	learn: 0.1040062	total: 591ms	remaining: 16.4ms
973:	learn: 0.1037605	total: 591ms	remaining: 15.8ms
974:	learn: 0.1034295	total: 592ms	remaining: 15.2ms
975:	learn: 0.1033488	total: 593ms	remaining: 14.6ms
976:	learn: 0.1031257	total: 593ms	remaining: 14ms
977:	learn: 0.1030126	total: 594ms	remaining: 13.4ms
978:	learn: 0.1028831	total: 594ms	remaining: 12.7ms
979:	learn: 0.1027515	total: 595ms	remaining: 12.1ms
980:	learn: 0.1026522	total: 596ms	remaining: 11.5ms
981:	learn: 0.1025260	total: 596ms	remaining: 10.9ms
982:	learn: 0.1024598	total: 597ms	remaining: 10.3ms
983:	learn: 0.1022288	total: 597ms	remaining: 9.71ms
984:	learn: 0.1020564	total: 598ms	remaining: 9.11ms
985:	learn: 0.1020242	total: 599ms	remaining: 8.5ms
986:	learn: 0.1017911	total: 599ms	remaining: 7.89ms
987:	learn: 0.1016684	total: 600ms	remaining: 7.29ms
988:	learn: 0.1014804	total: 600ms	remaining: 6.68ms
989:	learn: 0.1013950	total: 601ms	remaining: 6.07ms
990:	learn: 0.1013638	total: 602ms	remaining: 5.46ms
991:	learn: 0.1011769	total: 602ms	remaining: 4.86ms
992:	learn: 0.1011098	total: 603ms	remaining: 4.25ms
993:	learn: 0.1009741	total: 603ms	remaining: 3.64ms
994:	learn: 0.1007730	total: 604ms	remaining: 3.03ms
995:	learn: 0.1005491	total: 604ms	remaining: 2.43ms
996:	learn: 0.1003494	total: 605ms	remaining: 1.82ms
997:	learn: 0.1002632	total: 606ms	remaining: 1.21ms
998:	learn: 0.1001248	total: 606ms	remaining: 606us
999:	learn: 0.0999311	total: 607ms	remaining: 0us
<catboost.core.CatBoostRegressor at 0x792a76d56050>
```

```
Accuracy_1 = CBR.score(X_test, y_test)*100
print('Accuracy of model is:', Accuracy_1)

Prediction_1 = CBR.predict(X_test)
print(Prediction_1)

Actual_1 = (y_test)
print(Actual_1)
```

```
Accuracy of model is: 97.7253755502261
[12.32716206  6.59782523 13.26472875 18.21447185 17.00319775 10.48442375
 22.45942813 15.28896065 10.51806727 12.36433794 12.75486159  8.48176482
 17.15723688 21.44559999 21.28107718  7.75456869 16.22375255 14.42148902
  6.9032119  15.702565   14.79768261  9.74080604 15.30381673 10.66239862
 17.99611891 19.81857098 15.08223632 20.35922381  7.10309441 10.57457905
 12.17769707 14.85214819 25.0331309  21.57284174  9.77351595 19.71431768
 12.50785769 15.57364776  6.15594094 10.93630288 17.05153922 18.63559751
  9.20848666 14.56880965 12.40625671 12.69545414  3.97879822  7.18099368
 10.77644159  9.1653968 ]
16     12.5
8       4.8
146    13.2
70     18.3
64     18.0
5       7.2
93     22.2
62     15.7
186    10.3
```

```
115    12.6
88     12.9
44      8.5
89     16.7
137    20.8
39     21.5
91      7.3
152    16.6
160    14.4
76      6.9
154    15.6
162    14.9
2       9.3
37     14.7
79     11.0
184    17.6
193    19.6
169    15.0
124    19.7
172     7.6
145    10.3
177    11.7
123    15.2
61     24.2
111    21.8
129     9.7
133    19.6
165    11.9
45     14.9
189     6.7
51     10.7
194    17.3
68     18.9
136     9.5
156    15.3
164    11.9
135    11.6
78      5.3
```

```
from sklearn.metrics import mean_absolute_percentage_error
print('Error in model is :', mean_absolute_percentage_error(Actual_1, Prediction_1)*100)
```

```
Error in model is : 5.259365313403824
```

```
from xgboost import XGBRegressor

XGB = XGBRegressor()
```

```
XGB.fit(X_train, y_train)
```

```
▾                              XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

```
Accuracy_2 = XGB.score(X_test, y_test)*100
print('Accuracy of model is:', Accuracy_2)

Prediction_2 = XGB.predict(X_test)
print(Prediction_2)

Actual_2 = (y_test)
print(Actual_2)
```

```
 12.654001  14.6851    24.284903  22.780994  10.206216  20.024158
 12.88443   14.876586   6.6887784  9.672113  17.931408  19.522026
  9.102671  15.303094  12.227745  13.173177   3.3289135  6.444073
 10.461912   9.151075 ]
16     12.5
8       4.8
146    13.2
70     18.3
64     18.0
5       7.2
93     22.2
62     15.7
186    10.3
115    12.6
88     12.9
44      8.5
89     16.7
137    20.8
39     21.5
91      7.3
152    16.6
160    14.4
76      6.9
154    15.6
162    14.9
2       9.3
37     14.7
79     11.0
184    17.6
193    19.6
169    15.0
124    19.7
172     7.6
145    10.3
177    11.7
123    15.2
61     24.2
111    21.8
129     9.7
133    19.6
165    11.9
45     14.9
189     6.7
51     10.7
194    17.3
68     18.9
136     9.5
156    15.3
164    11.9
135    11.6
78      5.3
22      5.6
90     11.2
72      8.8
Name: Sales, dtype: float64
```

```
from sklearn.metrics import mean_absolute_percentage_error
print('Error in model is :', mean_absolute_percentage_error(Actual_2, Prediction_2)*100)
```

```
Error in model is : 4.99154252324751
```

```
from sklearn.ensemble import RandomForestRegressor

RF = RandomForestRegressor()
```

```
RF.fit(X_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```
Accuracy_3 = RF.score(X_test, y_test)*100
print('Accuracy of model is:', Accuracy_3)

Prediction_3 = RF.predict(X_test)
print(Prediction_3)
```

```
Actual_3 = (y_test)
print(Actual_3)
```

```
Accuracy of model is: 98.30514841449673
[12.395  5.288 12.811 18.196 16.977  7.705 22.463 15.458 10.565 12.756
 12.008  8.062 15.523 20.885 21.607  7.866 16.262 14.174  7.22  15.497
 15.029  8.56  13.851 11.054 16.772 19.342 14.49  20.618  6.991 10.552
 12.524 14.941 25.414 22.519 10.015 19.826 12.104 15.474  6.76  10.223
 16.633 19.756  8.401 14.119 12.939 12.267  3.816  6.417 11.245  8.546]
16     12.5
8       4.8
146    13.2
70     18.3
64     18.0
5       7.2
93     22.2
62     15.7
186    10.3
115    12.6
88     12.9
44      8.5
89     16.7
137    20.8
39     21.5
91      7.3
152    16.6
160    14.4
76      6.9
154    15.6
162    14.9
2       9.3
37     14.7
79     11.0
184    17.6
193    19.6
169    15.0
124    19.7
172     7.6
145    10.3
177    11.7
123    15.2
61     24.2
111    21.8
129     9.7
133    19.6
165    11.9
45     14.9
189     6.7
51     10.7
194    17.3
68     18.9
136     9.5
156    15.3
164    11.9
135    11.6
78      5.3
22      5.6
90     11.2
72      8.8
Name: Sales, dtype: float64
```

```
from sklearn.metrics import mean_absolute_percentage_error
print('Error in model is :', mean_absolute_percentage_error(Actual_3, Prediction_3)*100)
```

```
Error in model is : 4.609543632144995
```

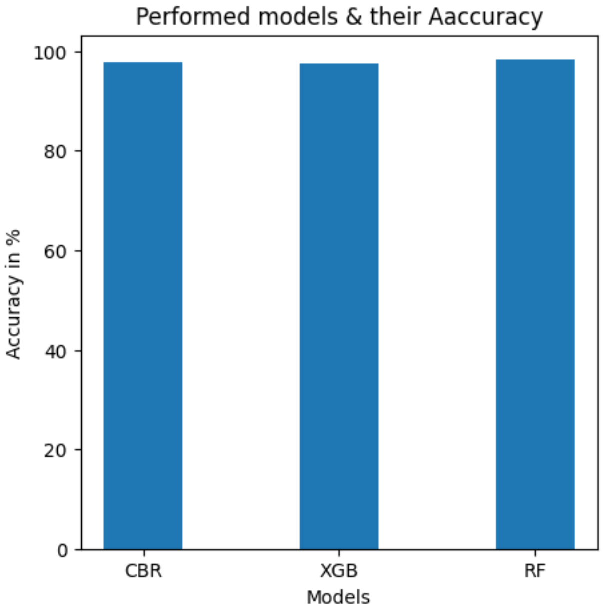
```
Result = pd.DataFrame({'Model' : ['CBR', 'XGB', 'RF'],
                       'Accuracy' : [Accuracy_1, Accuracy_2, Accuracy_3],
                       'Error' : [(mean_absolute_percentage_error(Actual_1, Prediction_1)*100), (mean_absolute_percentage_error(Actual_2, Pre

Result
```

```
      Model  Accuracy     Error
```

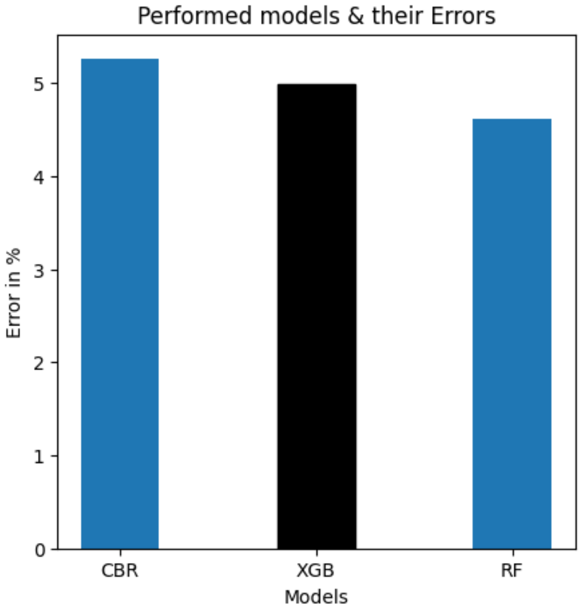
```
fig = plt.figure(figsize = (5, 5))

bars = plt.bar(Result['Model'], Result['Accuracy'],  width=0.4)
plt.xlabel('Models')
plt.ylabel('Accuracy in %')
plt.title('Performed models & their Aaccuracy')
plt.show()
```



```
fig = plt.figure(figsize = (5, 5))

bars = plt.bar(Result['Model'], Result['Error'],  width=0.4)
bars[1].set_color('black')
plt.xlabel('Models')
plt.ylabel('Error in %')
plt.title('Performed models & their Errors')
plt.show()
```



```
X_new = np.array([[254, 45, 10]])
#Prediction of the Species from the input vector
prediction = XGB.predict(X_new)
print("Sales: {}".format(prediction))
```

```
Sales: [24.653505]
```