ABHILASHA CHATTERJEE
DATA ANALYTICS & BUSINESS INTELLIGENCE BATCH 9

ASSIGNMENT ON REGRESION

PART 1

[8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[9]:
```
df = pd.read_csv(r"C:\Users\abc\Desktop\ASSINGMENTS\Marketing.csv")
```

[5]:
```
df.head()
```

[5]:

	S.NO	YOUTUBE	FACEBOOK	NEWSPAPER	SALES
0	1	276.12	45.36	83.04	26.52
1	2	53.40	47.16	54.12	12.48
2	3	20.64	55.08	83.16	11.16
3	4	181.80	49.56	70.20	22.20
4	5	216.96	12.96	70.08	15.48

[6]:
```
df.describe()
```

[6]:

	S.NO	YOUTUBE	FACEBOOK	NEWSPAPER	SALES
count	200.000000	200.000000	200.000000	200.000000	200.000000
mean	100.500000	176.451000	27.916800	36.664800	16.827000
std	57.879185	103.025084	17.816171	26.134345	6.260948
min	1.000000	0.840000	0.000000	0.360000	1.920000
25%	50.750000	89.250000	11.970000	15.300000	12.450000
50%	100.500000	179.700000	27.480000	30.900000	15.480000
75%	150.250000	262.590000	43.830000	54.120000	20.880000
max	200.000000	355.680000	59.520000	136.800000	32.400000

[11]:
```
df.isnull().sum()
```

[11]:
```
S.NO        0
YOUTUBE     0
FACEBOOK    0
```

```
NEWSPAPER    0
SALES        0
dtype: int64
```

a) Using Sklearn

[51]:
```
from sklearn.model_selection import train_test_split
```

[52]:
```
y = df['SALES']
x = df.drop(columns=['SALES'])
```

[53]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2,␣
↪random_state = 25)
```

[14]:
```
x_train
```

[14]:

	S.NO	YOUTUBE	FACEBOOK	NEWSPAPER
165	166	281.40	4.08	101.76
42	43	352.32	33.24	2.16
30	31	351.48	33.96	51.84
195	196	45.84	4.44	16.56
134	135	44.28	46.32	78.72
..	…	…	…	…
118	119	150.84	44.28	95.04
61	62	313.56	51.24	65.64
143	144	125.52	6.84	41.28
62	63	287.16	18.60	32.76
132	133	10.08	32.64	2.52

[160 rows x 4 columns]

[54]:
```
from sklearn.linear_model import LinearRegression
```

[55]:
```
model = LinearRegression()
```

[56]:
```
model.fit(x_train,y_train)
```

[56]:
```
LinearRegression()
```

[35]:
```
model.intercept_
```

[35]:
```
4.040822104087098
```

[36]:
```
model.coef_
```

[36]:
```
array([-0.00183379,  0.04604938,  0.18445335, -0.00573362])
```

[57]:
```
from sklearn.metrics import r2_score, mean_squared_error
```

[61]:
```
y_pred = model.predict(x_test)
```

[63]:
```
print(y_pred)
```

```
[14.70572646 16.80513414 12.31931164 22.17398948 27.57317974 12.50108957
 25.42252005 19.87418847 20.88769853 25.4430479  17.24121378 23.04411418
 17.91728064 21.42391996 16.61723106 23.98755809  8.12148785 21.74149594
 18.57540348 17.79524633 16.76713008 18.28654455 17.07092379 26.30055353
  4.47723586 18.17989546 14.30378837 11.8047825  20.87131182 23.57700771
 25.31460054 29.55548638 13.80671531 12.82527629 18.08619223 12.89763903
 10.74786331 21.33111351 16.93483966 21.75229696]
```

b) Using Statsmodel

[23]:
```
import statsmodels.api as sm
```

[24]:
```
x1 = sm.add_constant(x)
```

[25]:
```
model = sm.OLS(y,x1).fit()
```

[26]:
```
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  SALES   R-squared:                       0.897
Model:                            OLS   Adj. R-squared:                  0.895
Method:                 Least Squares   F-statistic:                     425.7
Date:                Thu, 16 Nov 2023   Prob (F-statistic):           3.94e-95
Time:                        20:16:08   Log-Likelihood:                -422.61
No. Observations:                 200   AIC:                             855.2
Df Residuals:                     195   BIC:                             871.7
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.6063      0.473      7.623      0.000       2.673       4.539
S.NO          -0.0007      0.003     -0.276      0.783      -0.006       0.004
YOUTUBE        0.0458      0.001     32.725      0.000       0.043       0.049
FACEBOOK       0.1884      0.009     21.784      0.000       0.171       0.205
NEWSPAPER     -0.0012      0.006     -0.210      0.834      -0.013       0.010
==============================================================================
Omnibus:                       60.267   Durbin-Watson:                   2.085
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              150.423
Skew:                          -1.325   Prob(JB):                     2.17e-33
```

```
Kurtosis:                       6.320   Cond. No.                         753.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

An R-squared and adjusted R-squared close to 1 indicate a good fit for our regression model,

Here R-squared of 0.897 means that approximately 89.7% of the variability in the dependent variable is explained by the independent variables in the model.

And the adjusted R-squared of 0.895 suggests that about 89.5% of the variability in the dependent variable is explained by the independent variables in your model.

The value of coeffecients are : -0.00183379, 0.04604938, 0.18445335, -0.00573362

It can be said that holding other variables constant, the dependent variable is expected to decrease by approximately -0.0018 , 0.0460, 0.1845 , -0.0057 units for a one-unit increase in each variable x1, x2, x3 and x4 respectively.

[40]:
```
from sklearn.metrics import r2_score, mean_squared_error
```

[70]:
```
r_squared = r2_score(y_test, y_pred)
print(f'R-squared: {r_squared}')
```

```
R-squared: 0.8720389628792302
```

[68]:
```
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error (MSE): {mse}')
```

```
Mean Squared Error (MSE): 4.805689866627322
```

[69]:
```
rmse = np.sqrt(mse)
print(f'Root Mean Squared Error (RMSE): {rmse}')
```

```
Root Mean Squared Error (RMSE): 2.1921883738920163
```

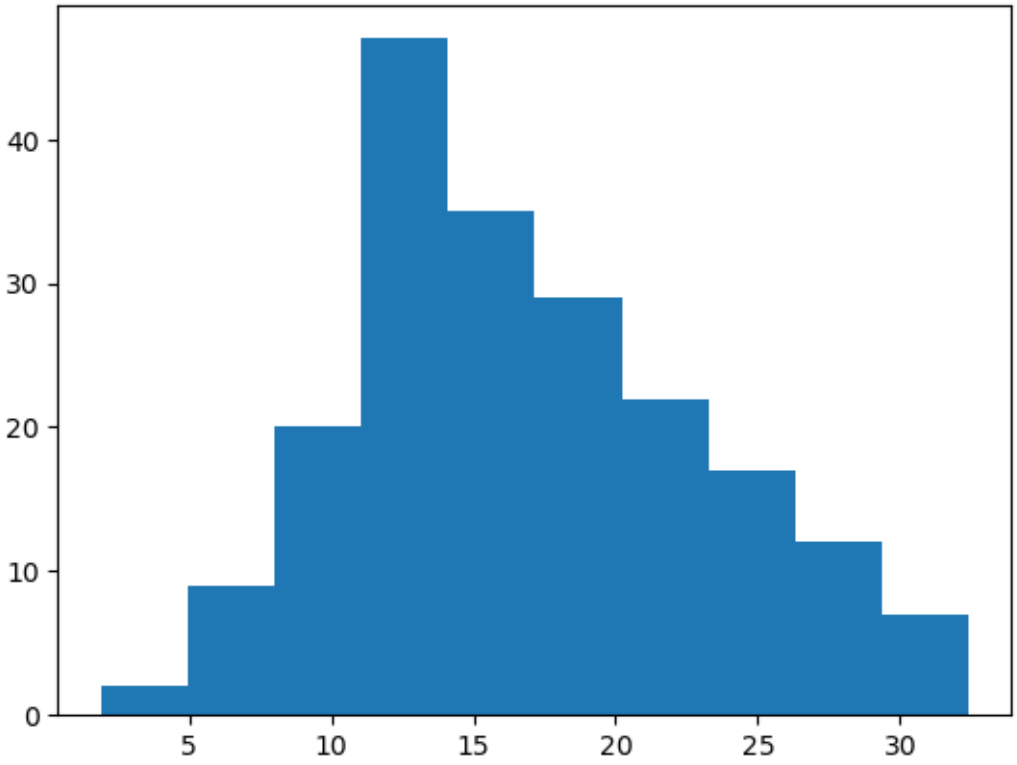
The R-squared value of 0.872 is considered quite good as higher the value of r2, the better it shows the goodness of the model.

The Mean Squared Error (MSE) is a measure of the average squared difference between the actual and predicted values in a regression model. Here it is approximately 4.81 meaning the model still shows some errors between actual and predicted values.

The Root Mean Squared Error (RMSE) is a measure of the average deviation between the predicted values and the actual values in a regression model. Here it means the model's predictions deviate by approximately 2.192 units from the actual values.

[9]:
```
plt.hist(df['SALES'])
```

[9]:
```
(array([ 2.,  9., 20., 47., 35., 29., 22., 17., 12.,  7.]),
 array([ 1.92 ,  4.968,  8.016, 11.064, 14.112, 17.16 , 20.208, 23.256,
        26.304, 29.352, 32.4  ]),
 <BarContainer object of 10 artists>)
```



PART 2

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
df = pd.read_csv(r"C:\Users\abc\Desktop\ASSINGMENTS\Diamonds.csv")
```

Variables in the data are : Carat, Cut, Depth, Table, Price, X, Y and Z

No. of rows = 53940

[7]:
```
len(df)
```

[7]:
```
53940
```

[14]:
```
df.describe()
```

[14]:
```
                S.NO         CARAT         DEPTH         TABLE         PRICE  \
count   53940.000000  53940.000000  53940.000000  53940.000000  53940.000000
mean    26970.500000      0.797940     61.749405     57.457184   3932.799722
std     15571.281097      0.474011      1.432621      2.234491   3989.439738
min         1.000000      0.200000     43.000000     43.000000    326.000000
25%     13485.750000      0.400000     61.000000     56.000000    950.000000
50%     26970.500000      0.700000     61.800000     57.000000   2401.000000
75%     40455.250000      1.040000     62.500000     59.000000   5324.250000
max     53940.000000      5.010000     79.000000     95.000000  18823.000000

                 X             Y             Z
count  53940.000000  53940.000000  53940.000000
mean       5.731157      5.734526      3.538734
std        1.121761      1.142135      0.705699
min        0.000000      0.000000      0.000000
25%        4.710000      4.720000      2.910000
50%        5.700000      5.710000      3.530000
75%        6.540000      6.540000      4.040000
max       10.740000     58.900000     31.800000
```

[3]:
```
from sklearn.model_selection import train_test_split
```

[4]:
```
y = df['CUT']
x = df.drop(columns=['CUT'])
```

[5]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.3,␣
↪random_state = 30)
```

[6]:
```
x_train
```

[6]:
```
        S.NO  CARAT  DEPTH  TABLE  PRICE     X     Y     Z
35511  35512   0.41   63.6   56.0    904  4.73  4.70  3.00
6782    6783   0.82   61.9   56.0   4113  5.99  6.02  3.72
42445  42446   0.57   61.2   57.0   1315  5.31  5.35  3.26
52574  52575   0.70   61.9   58.0   2536  5.71  5.67  3.52
19529  19530   1.01   61.2   58.0   8163  6.49  6.45  3.96
…        …    …      …      …     …     …     …     …
33268  33269   0.30   62.6   54.0    826  4.30  4.33  2.70
44845  44846   0.52   61.2   56.0   1625  5.19  5.37  3.20
48045  48046   0.51   61.9   55.0   1926  5.14  5.17  3.19
4517    4518   1.00   64.0   59.0   3634  6.29  6.24  4.01
38693  38694   0.37   61.1   57.0   1041  4.66  4.63  2.84

[37758 rows x 8 columns]
```

[7]:
```
from sklearn.linear_model import LinearRegression
```

[8]:
```
model = LinearRegression()
```

[9]:
```
model.fit(x_train,y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[9], line 1
----> 1 model.fit(x_train,y_train)

File E:\Users\abhilasha\jupiter\Lib\site-packages\sklearn\base.py:1151, in␣
 ↪_fit_context.<locals>.decorator.<locals>.wrapper(estimator, *args, **kwargs)
   1144     estimator._validate_params()
   1146 with config_context(
   1147     skip_parameter_validation=(
   1148         prefer_skip_nested_validation or global_skip_validation
   1149     )
   1150 ):
-> 1151     return fit_method(estimator, *args, **kwargs)

File E:\Users\abhilasha\jupiter\Lib\site-packages\sklearn\linear_model\_base.py
 ↪678, in LinearRegression.fit(self, X, y, sample_weight)
    674 n_jobs_ = self.n_jobs
    676 accept_sparse = False if self.positive else ["csr", "csc", "coo"]
--> 678 X, y = self._validate_data(
    679     X, y, accept_sparse=accept_sparse, y_numeric=True, multi_output=Tru
    680 )
    682 has_sw = sample_weight is not None
    683 if has_sw:

File E:\Users\abhilasha\jupiter\Lib\site-packages\sklearn\base.py:621, in␣
 ↪BaseEstimator._validate_data(self, X, y, reset, validate_separately,␣
 ↪cast_to_ndarray, **check_params)
    619         y = check_array(y, input_name="y", **check_y_params)
    620     else:
--> 621         X, y = check_X_y(X, y, **check_params)
    622     out = X, y
    624 if not no_val_X and check_params.get("ensure_2d", True):

File E:\Users\abhilasha\jupiter\Lib\site-packages\sklearn\utils\validation.py:
 ↪1163, in check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order,␣
 ↪copy, force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples␣
 ↪ensure_min_features, y_numeric, estimator)
   1143     raise ValueError(
   1144         f"{estimator_name} requires y to be passed, but the target y is␣
 ↪None"
   1145     )
   1147 X = check_array(
   1148     X,
```

```
   1149     accept_sparse=accept_sparse,
   (…)
   1160     input_name="X",
   1161 )
-> 1163 y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric,␣
 ↪estimator=estimator)
   1165 check_consistent_length(X, y)
   1167 return X, y

File E:\Users\abhilasha\jupiter\Lib\site-packages\sklearn\utils\validation.py:
 ↪1188, in _check_y(y, multi_output, y_numeric, estimator)
   1186     _ensure_no_complex_data(y)
   1187 if y_numeric and y.dtype.kind == "O":
-> 1188     y = y.astype(np.float64)
   1190 return y

ValueError: could not convert string to float: 'Good'
```

[ ]:
```
from sklearn.metrics import r2_score, mean_squared_error
```

[10]:
```
predicted_value = model.predict(x)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[10], line 1
----> 1 predicted_value = model.predict(x)

File E:\Users\abhilasha\jupiter\Lib\site-packages\sklearn\linear_model\_base.py
 ↪386, in LinearModel.predict(self, X)
    372 def predict(self, X):
    373     """
    374     Predict using the linear model.
    375
   (…)
    384         Returns predicted values.
    385     """
--> 386     return self._decision_function(X)

File E:\Users\abhilasha\jupiter\Lib\site-packages\sklearn\linear_model\_base.py
 ↪370, in LinearModel._decision_function(self, X)
    367 check_is_fitted(self)
    369 X = self._validate_data(X, accept_sparse=["csr", "csc", "coo"],␣
 ↪reset=False)
--> 370 return safe_sparse_dot(X, self.coef_.T, dense_output=True) + self.
 ↪intercept_
```

```
AttributeError: 'LinearRegression' object has no attribute 'coef_'
```