ASSIGNMENT 1 – LINEAR REGRESSION

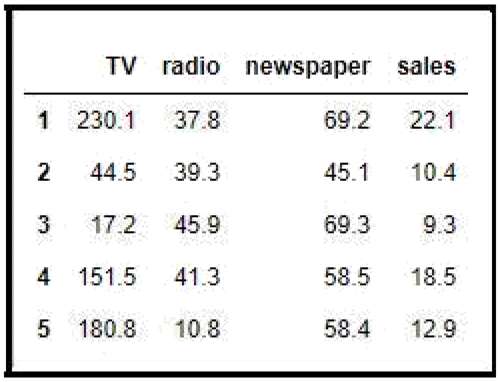
Submitted by – Kartik Nagras

Dataset Explanation

This dataset consists of 200 observations and 4 columns namely ‘TV’, ’Radio’ ’Newspaper’ and ‘Sales’. The first three columns [namely ‘TV’, ’Radio’ & ’Newspaper’] consists of the amount of money spend on advertising for a product through each of the respective medium and ‘Sales’ column consist of the amount of money generated through sale of that product.

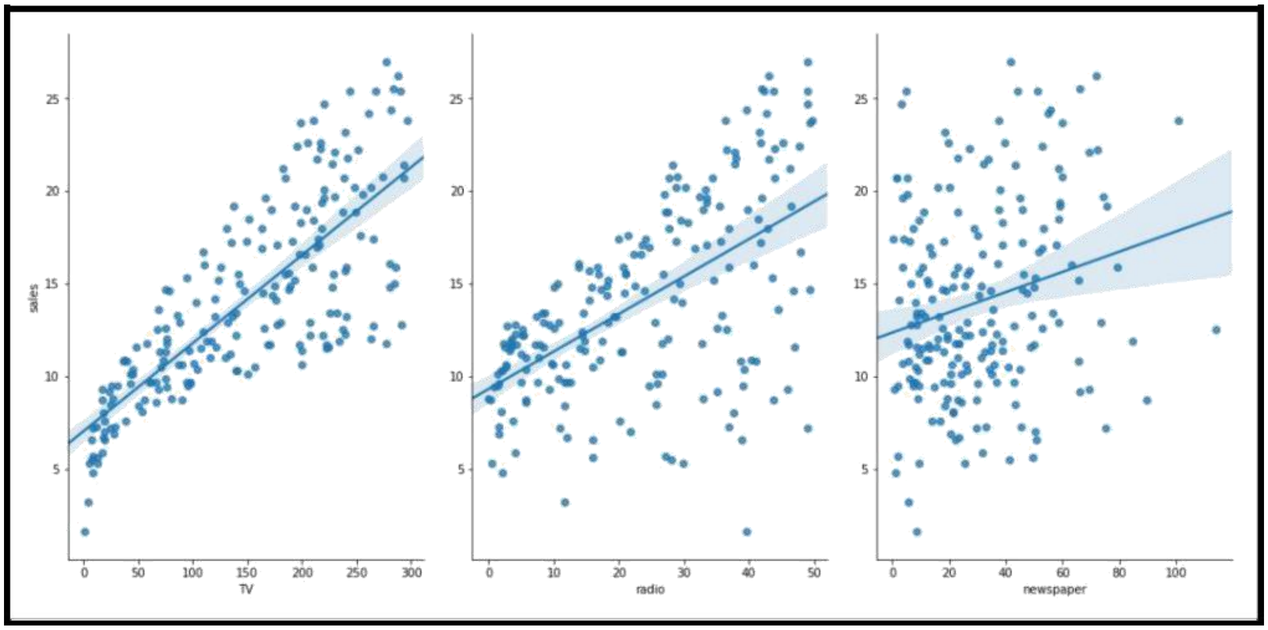
Data Extraction and Preprocessing

To import the data, we use the Pandas library. The head & shape method tell us that the dataset has three features aka independent variables namely ‘TV’, ’Radio’ & ’Newspaper’ and ‘Sales’ is the response aka dependent variable in this dataset.



	TV	radio	newspaper	sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

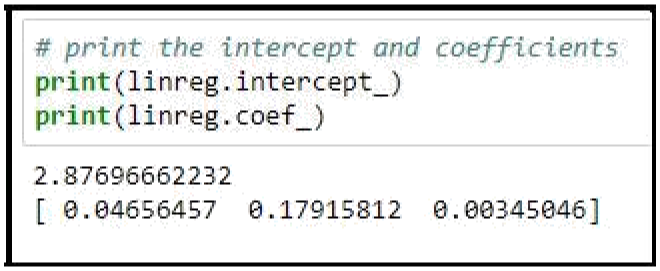
Consequently, Linear regression can be applied to this dataset to predict the sales based on the marketing expenditure via ‘TV’, ’Radio’ & ’Newspaper’. Next, we visualize the relationship between the features and the response variable using scatterplots, created using the pairplot function in seaborn library. This helps in plotting different correlation graphs together for better analysis. From this we see that marketing expenditure via ‘TV’ has the highest correlation with sales and marketing expenditure via ‘Newspaper’ as medium has the lowest correlation with sales.



Applying Linear Regression

Then we segregate the data such as., all independent variables will be stored in X and the dependent variable 'Sales' will be stored in y. Now we want to split our data set into training dataset and test data set so we use cross_validation with test_size = 0.25 (default) meaning 25% of our data set will be used for testing, and we need to specify the random test as 0 (we can use any integer for random_state it does not affect the result it will give you result with different integers) otherwise every time we get different result. Very important is to remember that scikit learn algorithms work with numpy arrays. Next, we print the shapes of all the testing and training datasets for validation sake.

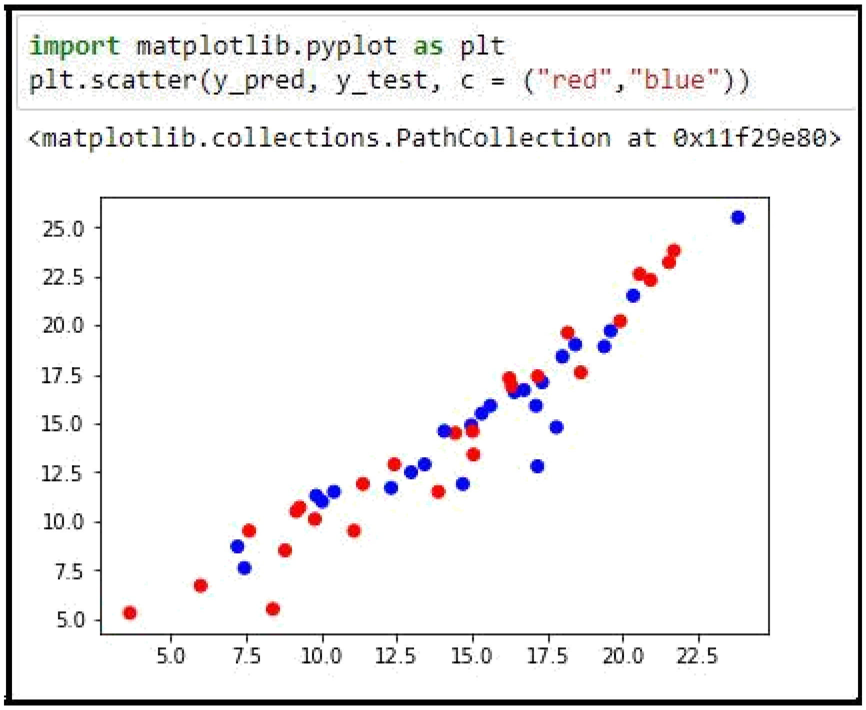
Next, we apply a Linear Regression algorithm from the sci-kit package in python(sklearn) on the training datasets and print the intercept and coefficient we get.



```
# print the intercept and coefficients
print(linreg.intercept_)
print(linreg.coef_)
```

```
2.87696662232
[ 0.04656457  0.17915812  0.00345046]
```

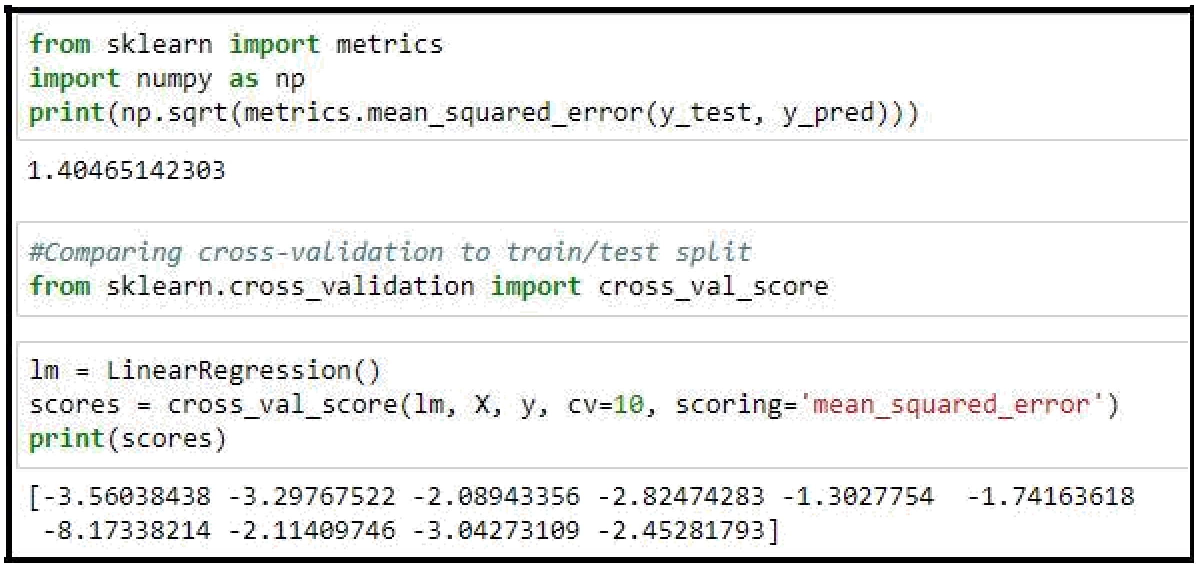
Further steps help in checking the validity of the applied algorithm on our dataset. We first predict the test data (X_test) through our trained Linear Regression model and compare the output (y_pred) with the testing data which has corresponding values of sales (y_test). We plotted the corresponding values in a scatterplot using matplotlib library and calculated an **accuracy score of 91.5%** of our model.



```
import matplotlib.pyplot as plt
plt.scatter(y_pred, y_test, c = ("red","blue"))
```

```
<matplotlib.collections.PathCollection at 0x11f29e80>
```

We get an RMSE of 1.404 from our model, and this was done using the metrics class imported from the scikit library. To further check the efficacy of the model applied we applied a K-fold cross-validation on the dataset which will repeat the Linear Regression on the same dataset 10 times (cv=10) but this time it would generate a different training and testing dataset in each iteration and thus provide us a mean square error value for each iteration.



```
from sklearn import metrics
import numpy as np
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

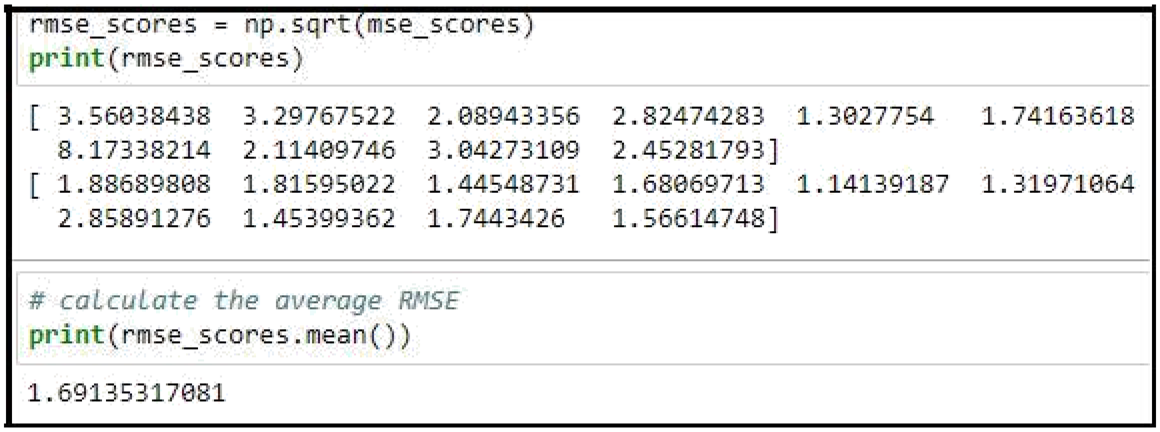
```
1.40465142303
```

```
#Comparing cross-validation to train/test split
from sklearn.cross_validation import cross_val_score
```

```
lm = LinearRegression()
scores = cross_val_score(lm, X, y, cv=10, scoring='mean_squared_error')
print(scores)
```

```
[-3.56038438 -3.29767522 -2.08943356 -2.82474283 -1.3027754  -1.74163618
 -8.17338214 -2.11409746 -3.04273109 -2.45281793]
```

Next, we use the sqrt method made available from the numpy library and calculate the square root of all the mean square errors. This value is averaged out and comes to about 1.69 which is significantly higher than what we got initially and thus provides further confidence to the quality of the Linear Regression model we generated for the dataset.



```
rmse_scores = np.sqrt(mse_scores)
print(rmse_scores)
```

```
[ 3.56038438  3.29767522  2.08943356  2.82474283  1.3027754   1.74163618
  8.17338214  2.11409746  3.04273109  2.45281793]
[ 1.88689808  1.81595022  1.44548731  1.68069713  1.14139187  1.31971064
  2.85891276  1.45399362  1.7443426   1.56614748]
```

```
# calculate the average RMSE
print(rmse_scores.mean())
```

```
1.69135317081
```