In [ ]:

Project: Linear Regression analysis on "advertising dataset" I have downloaded the "Advertising.csv" file from "An Introduction to Statistical Learning with Applications in R. This project aims to show my skills in data analysis and linear regression modeling using Python and scikit-learn. My work is for educational purposes.

In [218…
```
# Python_Machine_Learning_Linear_Regression_project_02

# Cross Validation
```

# Make Training and Testing datasets from the "Advertising.csv" file.

In [221…
```
# Read the .csv file
import pandas as pd
df = pd.read_csv("Advertising.csv")

#  Choose X featurers
X = df.drop('sales', axis = 1)

# Choose y response " sales column)
y = df['sales']


# Split the X and y into Train and Test parts

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 101)

# Scale the X dataset.
# load StandardScaler
from sklearn.preprocessing import StandardScaler
scaler= StandardScaler()

#  Scalar fit on the training and testing X Data.
scaler.fit(X_train)

X_train = scaler.transform(X_train)

X_test = scaler.transform(X_test)
```

In [223…
```
# Create a Model
# Use Ridge Model

# Load model
from sklearn.linear_model import Ridge

# Make a model
model = Ridge(alpha = 10)     # Choose alpha

# Fit the above Model on X_Train, and y_train dataset
model.fit(X_train, y_train)

# Predict y values on the X_test data using the fitted model.
y_predicted = model.predict(X_test)

# find mean squared error from y_test, and y_pred values
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_predicted)
```

Out[223… 2.3043155882835635

In [225…
```
# Adjust parameters as Necessary and repeat just last two steps.

# Now repeat the above steps using different alpha values.

model_10 = Ridge(alpha = 0.1)
model_10.fit(X_train, y_train)
y_predicted_10 = model_10.predict(X_test)
mean_squared_error(y_test, y_predicted_10)
```

Out[225… 2.1096939229756075

In [227…
```
# To minimize the data leakage, dataset is divided into three paarts (train, validation, and test). test data is used for final model test.

# First split dataset
X_train, X_test1, y_train, y_test1 = train_test_split(X, y, test_size = 0.25, random_state = 101)

# second split dataset.
X_train2, X_test, y_test2, y_test = train_test_split(X_test1, y_test1, test_size = 0.4, random_state = 101)

# Use StandardScaler

from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()

# fit scalar on X_train data
scaler.fit(X_train)

# transform for three datasets
X_train = scaler.transform(X_train)
X_train2 = scaler.transform(X_train2)
X_test = scaler.transform(X_test)

# Make a linear model
from sklearn.linear_model import Ridge
model1 = Ridge(alpha = 100)

# fit train datasets.
model1.fit(X_train, y_train)

# predict on X_train2
y_test2_predict = model.predict(X_train2)

# check error
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test2, y_test2_predict)

# Choose alpha = 0.1

model2 = Ridge(alpha = 0.1)

# # fit train datasets.

model2.fit(X_train, y_train)

# prediction
second_predict = model2.predict(X_train2)

# check error
mean_squared_error(y_test2, second_predict)
print(mean_squared_error(y_test2, second_predict))

# # Final performance if you satisfy at alpha = 0.1
y_test_prediction = model2.predict(X_test)
print(y_test_prediction)
mean_squared_error(y_test, y_test_prediction)
```

```
1.9391609593785695
[ 8.13368275 18.42064041 18.21958416  9.93548647 24.7130477  19.55515613
 21.42209051  8.89562661 10.11448833 16.19293195 15.44979271  7.83548211
 17.23369351  9.97825676  7.42449941 17.00033491  9.64243739 10.72810643
  9.88551126  7.67363797]
```

Out[227… 2.3654933683711636

In [229…
```
# Use K-Fold cross calidation for error analysis in the ("Advertising.csv").

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.75, random_state = 101)

from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()

scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

# Make a linear model(Ridge)
from sklearn.linear_model import Ridge
model_1 = Ridge(alpha = 100)

from sklearn.model_selection import cross_val_score

cv_scors = cross_val_score(model_1, X_train,y_train, scoring ='neg_mean_squared_error', cv = 7) # k-fole (cv) value = 7.
print(cv_scors)

# Absolute mean of cv_scors

print(abs(scores.mean()))

# Make another model
model_2 = Ridge(alpha = 1000)

scor_2 = cross_val_score(model_2, X_train,y_train, scoring ='neg_mean_squared_error', cv = 7) # k-fole (cv) value is 7.
print(scor_2)

print(abs(scor_2.mean()))

# Use model to fit all train data

model_final= Ridge(alpha = 1)
model_final.fit(X_train, y_train)
y_final_test_prediction  = model_final.predict(X_test)
mean_squared_error(y_test, y_final_test_prediction)
print(mean_squared_error(y_test, y_final_test_prediction))
```

```
[ -1.59725436 -17.6553551  -25.94686869  -7.26031918 -13.86382377
 -17.17616644  -9.27138728]
fit_time                        0.000801
score_time                      0.000901
test_neg_mean_squared_error     3.323018
test_neg_mean_absolute_error    1.308467
dtype: float64
[ -3.54007101 -27.85277703 -36.30204922 -10.95084931 -20.36331675
 -26.55120437 -13.73958657]
```

```
19.89997918056674
3.502569470564967
```

In [ ]: