





# **PROJECT REPORT**

**PROBLEM STATEMENT:** Prepare an ML model which can predict the profit value of a company if the value of its R&D Spend, Administration cost and Marketing Spend are given.

**SUBMITTED BY:** MOHAMMAD ADIL

## **TABLE OF CONTENTS:**

	ABSTRACT
	INTRODUCTION
	METHOD USED
	RESULTS

## **ABSTRACT:**

Supervised learning is a machine learning algorithm of inferring a function from labelled training data, and one of the types of supervised learning is Regression. In case of Regression problems we try to predict a value that depends on many variables (features).

We train our data using various Regression algorithms and then we check against the test set and compare using various evaluation metrics like root mean square error,  $r^2$  score, variance score etc.

For the given problem statement we are given a dataset of 50 startups, we are going to use various regression algorithms and comparing them against regression metrics to find out which one is better predictor of the profit values of the startups.

## **INTRODUCTION:**

For the dataset given, the profit value of the startups depends on three features namely R&D Spend, Administration Cost and Marketing Spend.

We will split our data into train and test set before applying any regression algorithms. After splitting, we will train the model on the train set and then use the trained model on the test set to find the evaluation metrics.

Here we have used seven regression algorithms: Linear Regression, Ridge Regression, Stochastic Gradient Descent, Random Forest, Decision Tree, Xgboost, Gradient Boosting.

## METHODS USED:

```
# Importing the required libraries
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline
```

```
# Loading the dataset using pandas
```

```
data = pd.read_csv('50_Startups.csv')
data.head()
```

	R&D Spend	Administration	Marketing Spend	Profit
0	165349.20	136897.80	471784.10	192261.83
1	162597.70	151377.59	443898.53	191792.06
2	153441.51	101145.55	407934.54	191050.39
3	144372.41	118671.85	383199.62	182901.99
4	142107.34	91391.77	366168.42	166187.94

Some statistical information about our dataset:

```
: data.describe()
```

```
:
```

	R&D Spend	Administration	Marketing Spend	Profit
count	50.000000	50.000000	50.000000	50.000000
mean	73721.615600	121344.639600	211025.097800	112012.639200
std	45902.256482	28017.802755	122290.310726	40306.180338
min	0.000000	51283.140000	0.000000	14681.400000
25%	39936.370000	103730.875000	129300.132500	90138.902500
50%	73051.080000	122699.795000	212716.240000	107978.190000
75%	101602.800000	144842.180000	299469.085000	139765.977500
max	165349.200000	182645.560000	471784.100000	192261.830000

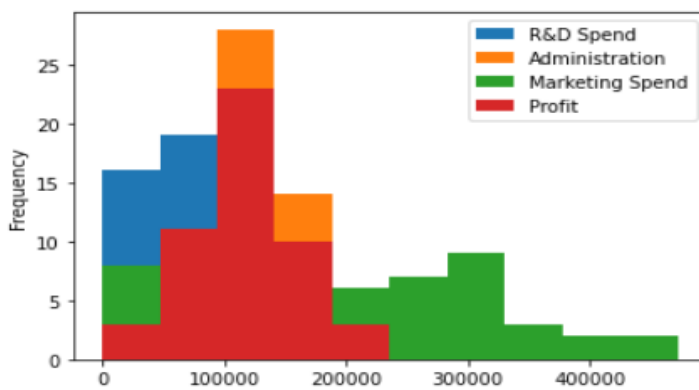
## Visualizing the dataframe

```
data.plot(x="Profit", y=["R&D Spend", "Administration", "Marketing Spend"])
plt.show()
```



```
data.plot(kind='hist')
```

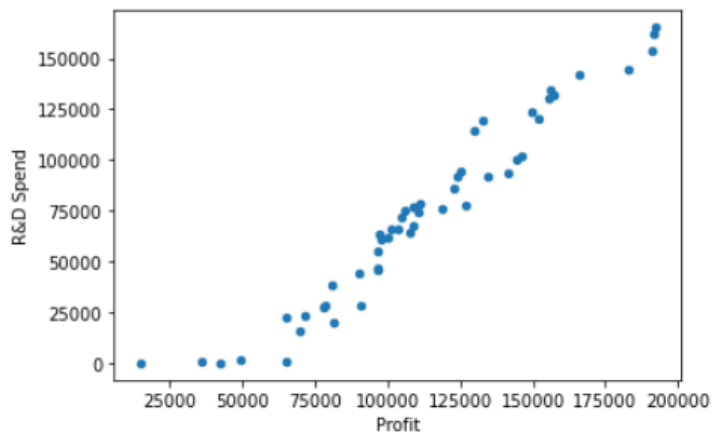
<AxesSubplot:ylabel='Frequency'>



## checking for Correlation

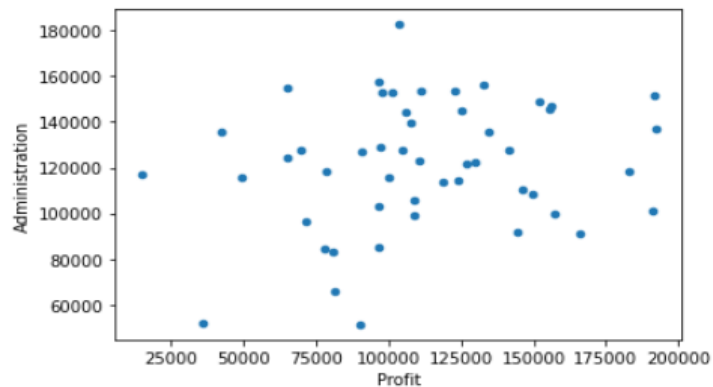
```
: data.plot(x='Profit',y='R&D Spend', kind='scatter')
```

: <AxesSubplot:xlabel='Profit', ylabel='R&D Spend'>



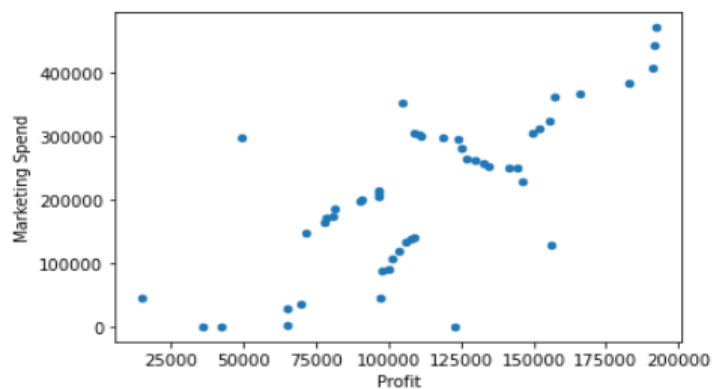
```
data.plot(x='Profit',y='Administration', kind='scatter')
```

```
<AxesSubplot:xlabel='Profit', ylabel='Administration'>
```



```
data.plot(x='Profit',y='Marketing Spend', kind='scatter')
```

```
<AxesSubplot:xlabel='Profit', ylabel='Marketing Spend'>
```



After visualization and checking for any correlation, we now scale our data using standard scaler and then split our data into training and test set.

## Scaling and Splitting the dataset

```
# Splitting the data into features and labels
```

```
x = data.drop('Profit', axis=1) #features  
y = data.Profit #labels
```

```
# Scaling the data using StandardScaler before applying regression algorithms
```

```
from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler()  
x_scaled = scaler.fit_transform(x)  
x_scaled
```

```
# Splitting the data into training and testing set using train_test_Split

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_scaled , y, test_size=0.2, random_state=21)
```

```
# Converting into numpy arrays
```

```
x_train = np.array(x_train)
x_test = np.array(x_test)
y_train = np.array(y_train)
y_test = np.array(y_test)
```

Now, we will apply various regression algorithms:

```
# Importing Evaluation metrics
```

```
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_squared_log_error
from sklearn.metrics import max_error
from sklearn.metrics import explained_variance_score
```

## LINEAR REGRESSION

```
from sklearn import linear_model
reg = linear_model.LinearRegression()

reg.fit(x_train,y_train)
y_pred_reg = reg.predict(x_test)

print('r2_score:',r2_score(y_test, y_pred_reg))
print('mean_absolute_error:',mean_absolute_error(y_test, y_pred_reg))
print('mean_squared_error:',mean_squared_error(y_test, y_pred_reg))
print('mean_squared_log_error:',mean_squared_log_error(y_test, y_pred_reg))
print('max_error:',max_error(y_test, y_pred_reg))
print('explained_variance_score:',explained_variance_score(y_test, y_pred_reg))
reg.score(x_test, y_test)
```

```
r2_score: 0.9658747497132996
mean_absolute_error: 5912.410683972493
mean_squared_error: 60563681.1063965
mean_squared_log_error: 0.005440873357421517
max_error: 18275.380249433772
explained_variance_score: 0.9669037542256351
0.9658747497132996
```

## RIDGE REGRESSION

```
from sklearn import linear_model
rid = linear_model.Ridge(alpha=.5)

rid.fit(x_train,y_train)
y_pred_rid = rid.predict(x_test)

print('r2_score:',r2_score(y_test, y_pred_rid))
print('mean_absolute_error:',mean_absolute_error(y_test, y_pred_rid))
print('mean_squared_error:',mean_squared_error(y_test, y_pred_rid))
print('mean_squared_log_error:',mean_squared_log_error(y_test, y_pred_rid))
print('max_error:',max_error(y_test, y_pred_rid))
print('explained_variance_score:',explained_variance_score(y_test, y_pred_rid))
```

```
rid.score(x_test, y_test)

r2_score: 0.9659490034384169
mean_absolute_error: 6045.635616587119
mean_squared_error: 60431899.54021923
mean_squared_log_error: 0.005509444257690543
max_error: 17886.83717910372
explained_variance_score: 0.9670581826801208
0.9659490034384169
```

## Stochastic Gradient Descent

```
from sklearn.linear_model import SGDRegressor
sgd = SGDRegressor(max_iter=1000, tol=1e-3)

sgd.fit(x_train, y_train)
y_pred_sgd = sgd.predict(x_test)

print('r2_score:', r2_score(y_test, y_pred_sgd))
print('mean_absolute_error:', mean_absolute_error(y_test, y_pred_sgd))
print('mean_squared_error:', mean_squared_error(y_test, y_pred_sgd))
print('mean_squared_log_error:', mean_squared_log_error(y_test, y_pred_sgd))
print('max_error:', max_error(y_test, y_pred_sgd))

print('explained_variance_score:', explained_variance_score(y_test, y_pred_sgd))

sgd.score(x_test, y_test)

r2_score: 0.9658243169513142
mean_absolute_error: 5936.562675059979
mean_squared_error: 60653186.49283413
mean_squared_log_error: 0.0054557783110263025
max_error: 18226.277877468703
explained_variance_score: 0.9668587688797976

0.9658243169513142
```

## RANDOM FOREST

```
from sklearn.ensemble import RandomForestRegressor
regr = RandomForestRegressor(max_depth=2, random_state=0)

regr.fit(x_train, y_train)
y_pred_regr = regr.predict(x_test)

print('r2_score:', r2_score(y_test, y_pred_regr))
print('mean_absolute_error:', mean_absolute_error(y_test, y_pred_regr))
print('mean_squared_error:', mean_squared_error(y_test, y_pred_regr))
print('mean_squared_log_error:', mean_squared_log_error(y_test, y_pred_regr))
print('max_error:', max_error(y_test, y_pred_regr))
print('explained_variance_score:', explained_variance_score(y_test, y_pred_regr))

regr.score(x_test, y_test)

r2_score: 0.9339275285069357
mean_absolute_error: 9898.896327121965
mean_squared_error: 117261911.92147665
mean_squared_log_error: 0.012274337749154781
max_error: 15589.249342461611
explained_variance_score: 0.9369511707774676

0.9339275285069357
```

## Decision Tree

```
from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor(max_depth=5)

dtr.fit(x_train, y_train)
y_pred_dtr = dtr.predict(x_test)

print('r2_score:', r2_score(y_test, y_pred_dtr))
print('mean_absolute_error:', mean_absolute_error(y_test, y_pred_dtr))
print('mean_squared_error:', mean_squared_error(y_test, y_pred_dtr))
print('mean_squared_log_error:', mean_squared_log_error(y_test, y_pred_dtr))
print('max_error:', max_error(y_test, y_pred_dtr))
print('explained_variance_score:', explained_variance_score(y_test, y_pred_dtr))

dtr.score(x_test, y_test)

r2_score: 0.9669314460457668
mean_absolute_error: 5867.402800000003
mean_squared_error: 58688312.59867342
mean_squared_log_error: 0.006674337988918017
max_error: 18383.214999999997
explained_variance_score: 0.9670650546607749

0.9669314460457668
```

## Xgboost

```
import xgboost as xg
xgb = xg.XGBRegressor(objective='reg:linear',n_estimators=10, seed=123)

xgb.fit(x_train, y_train)
y_pred_xgb = xgb.predict(x_test)

print('r2_score:',r2_score(y_test, y_pred_xgb))
print('mean_absolute_error:',mean_absolute_error(y_test, y_pred_xgb))
print('mean_squared_error:',mean_squared_error(y_test, y_pred_xgb))
print('mean_squared_log_error:',mean_squared_log_error(y_test, y_pred_xgb))
print('max_error:',max_error(y_test, y_pred_xgb))
print('explained_variance_score:',explained_variance_score(y_test, y_pred_xgb))

xgb.score(x_test, y_test)
```

[16:20:52] WARNING: c:\users\dev-admin\croot2\xgboost-split\_1675461376218\work\src\objec  
is now deprecated in favor of reg:squarederror.  
r2\_score: 0.9065323404608379  
mean\_absolute\_error: 10748.275558593752  
mean\_squared\_error: 165881436.13695917  
mean\_squared\_log\_error: 0.10907490328518624  
max\_error: 27036.471210937503  
explained\_variance\_score: 0.9189452996108837  
0.9065323404608379

## Gradient Boosting

```
from sklearn.ensemble import GradientBoostingRegressor
gbr = GradientBoostingRegressor(n_estimators = 200, max_depth = 1, random_state = 1)

gbr.fit(x_train, y_train)
y_pred_gbr = gbr.predict(x_test)

print('r2_score:',r2_score(y_test, y_pred_gbr))
print('mean_absolute_error:',mean_absolute_error(y_test, y_pred_gbr))
print('mean_squared_error:',mean_squared_error(y_test, y_pred_gbr))
print('mean_squared_log_error:',mean_squared_log_error(y_test, y_pred_gbr))
print('max_error:',max_error(y_test, y_pred_gbr))
print('explained_variance_score:',explained_variance_score(y_test, y_pred_gbr))

gbr.score(x_test, y_test)
```

r2\_score: 0.9300039372876133  
mean\_absolute\_error: 9061.28803856215  
mean\_squared\_error: 124225293.15392172  
mean\_squared\_log\_error: 0.06372817366448633  
max\_error: 22704.894079956455  
explained\_variance\_score: 0.9302095424708565  
0.9300039372876133

## RESULTS:

After training the different regression models and finding the evaluation metrics for each of them, we find that the prediction score is best for Decision tree and it's approximately close with Stochastic gradient descent and ridge/Linear Regression.

So, for our given dataset we can use any of the above three regression models to predict the profit values for the startups.