In [89]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
import statsmodels.api as sm
from statsmodels.graphics.regressionplots import influence_plot
import warnings
warnings.filterwarnings('ignore')
```

1. import dataset

In [90]:

```
startup_data=pd.read_csv('50_Startups (1).csv')
startup_data.head()
```

Out[90]:

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

2. Initial analysis

In [91]:

```
startup_data.describe()
```

Out[91]:

	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

In [92]:

```
print('There are ',startup_data.shape[0],'rows and ',startup_data.shape[1],'columns in the
```

```
There are  50 rows and  5 columns in the dataset.
```

In [93]:

```
print('There are',startup_data.duplicated().sum(),'duplicate values in the dateset.') #usin
```

```
There are 0 duplicate values in the dateset.
```

In [94]:

```
startup_data.isnull().sum()
```

Out[94]:

```
R&D Spend          0
Administration     0
Marketing Spend    0
State              0
Profit             0
dtype: int64
```

In [95]:

```
startup_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   R&D Spend        50 non-null     float64
 1   Administration   50 non-null     float64
 2   Marketing Spend  50 non-null     float64
 3   State            50 non-null     object 
 4   Profit           50 non-null     float64
dtypes: float64(4), object(1)
memory usage: 2.1+ KB
```

In [96]:

```
c=startup_data.corr()
c
```

Out[96]:

	R&D Spend	Administration	Marketing Spend	Profit
R&D Spend	1.000000	0.241955	0.724248	0.972900
Administration	0.241955	1.000000	-0.032154	0.200717
Marketing Spend	0.724248	-0.032154	1.000000	0.747766
Profit	0.972900	0.200717	0.747766	1.000000

Inference: We can see that all three columns have a direct relationship with the profit, which is our target variable.

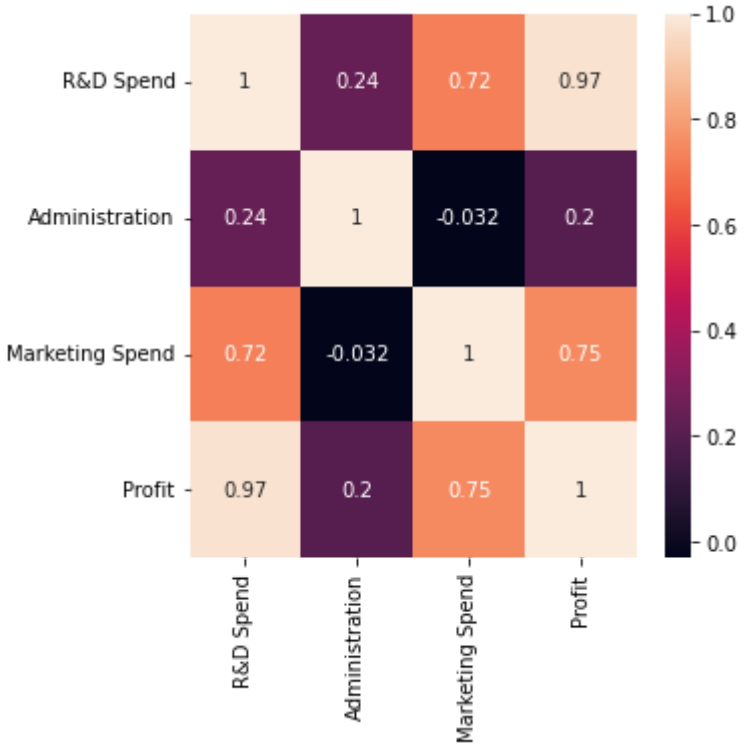
3.EDA on dataset

In [97]:

```
sns.heatmap(c,annot=True)
plt.show
```

Out[97]:

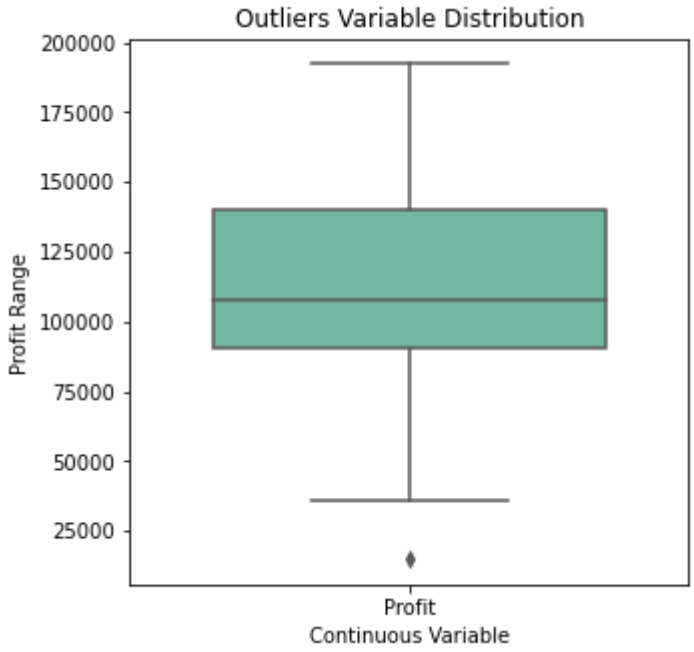
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Outliers detection in the target variable

In [98]:

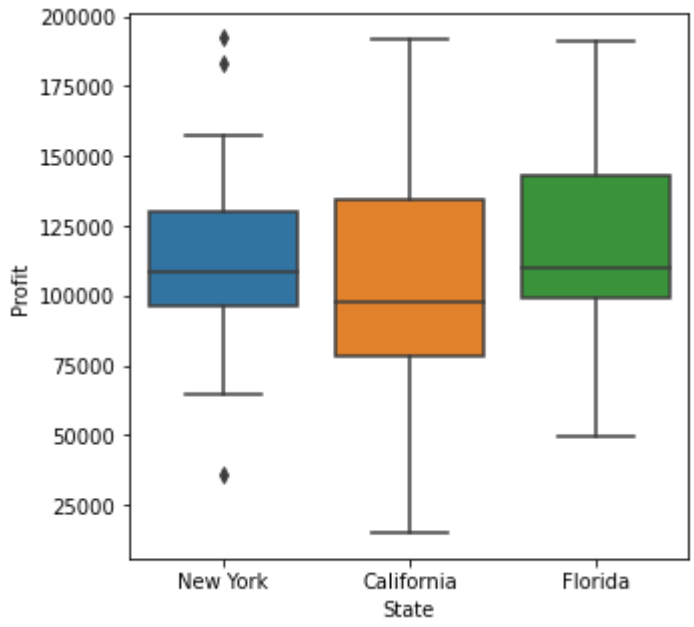
```
outliers = ['Profit']
plt.rcParams['figure.figsize'] = [5,5]
sns.boxplot(data=startup_data[outliers], orient="v", palette="Set2" , width=0.7) # orient =
                                                                                  # orient = "h"
plt.title("Outliers Variable Distribution")
plt.ylabel("Profit Range")
plt.xlabel("Continuous Variable")

plt.show()
```



Inference: While looking at the boxplot we can see the outliers in the profit(target variable), but the amount of data is not much (just 50 entries) so it won't create much negative impact.

In [99]:

```
sns.boxplot(x = 'State', y = 'Profit', data =startup_data)
plt.show()
```



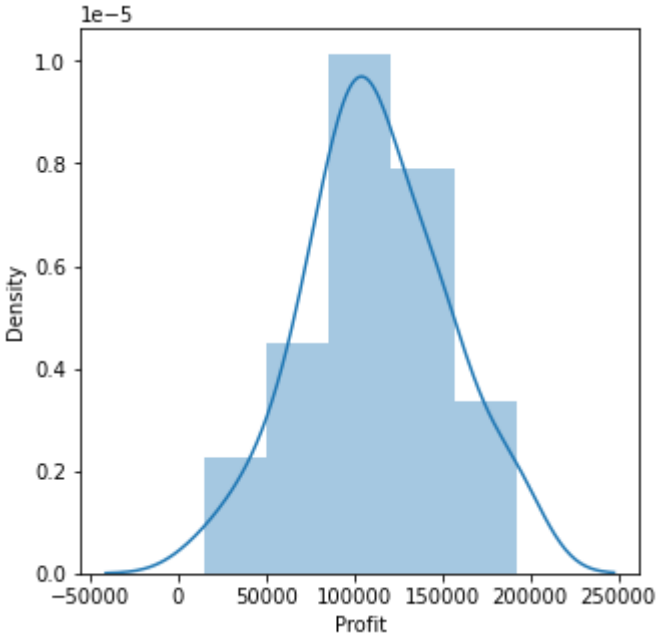
Insights:

1. All outliers presented are in New York.
2. The startups located in California we can see the maximum profits and maximum loss.

Histogram on profit

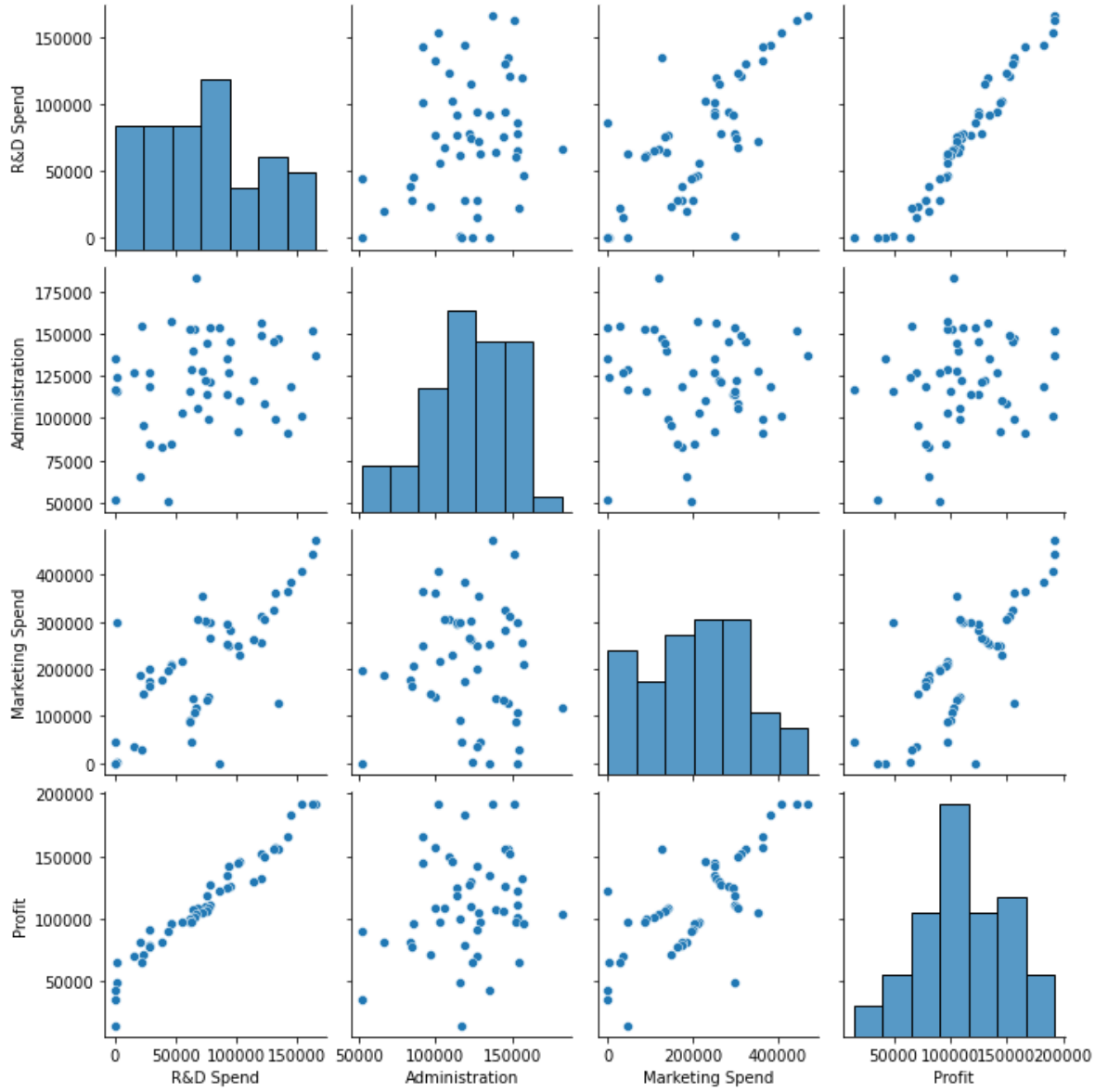
In [100]:

```
sns.distplot(startup_data['Profit'],bins=5,kde=True)
plt.show()
```



In [101]:

```
sns.pairplot(startup_data)
plt.show()
```



Inference:

1. As I can see in the pair pot, Research and development are directly proportional to the investment that we can do.
2. The marketing spend seems to be directly proportional (though a little bit outliers are there) with the profit.
3. There is no relationship between the second column and profit i.e. our target column.

4.Model Development

In [102]:

```
#splitting startup_data in Dependent & Independent Variables
x=startup_data.iloc[:, :-1].values
y=startup_data.iloc[:, 4].values
```

Label Encoder

In [103]:

```
from sklearn.preprocessing import LabelEncoder
```

In [104]:

```
labelencoder = LabelEncoder()
x[:, 3] = labelencoder.fit_transform(x[:, 3])
x1 = pd.DataFrame(x)
x1.head()
```

Out[104]:

	0	1	2	3
0	165349.2	136897.8	471784.1	2
1	162597.7	151377.59	443898.53	0
2	153441.51	101145.55	407934.54	1
3	144372.41	118671.85	383199.62	2
4	142107.34	91391.77	366168.42	1

5.Spliting the data into training and testing data

In [105]:

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.7,random_state=0)
x_train
```

Out[105]:

```
array([[130298.13, 145530.06, 323876.68, 1],
       [119943.24, 156547.42, 256512.92, 1],
       [1000.23, 124153.04, 1903.93, 2],
       [542.05, 51743.15, 0.0, 2],
       [65605.48, 153032.06, 107138.38, 2],
       [114523.61, 122616.84, 261776.23, 2],
       [61994.48, 115641.28, 91131.24, 1],
       [63408.86, 129219.61, 46085.25, 0],
       [78013.11, 121597.55, 264346.06, 0],
       [23640.93, 96189.63, 148001.11, 0],
       [76253.86, 113867.3, 298664.47, 0],
       [15505.73, 127382.3, 35534.17, 2],
       [120542.52, 148718.95, 311613.29, 2],
       [91992.39, 135495.07, 252664.93, 0],
       [64664.71, 139553.16, 137962.62, 0],
       [131876.9, 99814.71, 362861.36, 2],
       [94657.16, 145077.58, 282574.31, 2],
       [28754.33, 118546.05, 172795.67, 0],
       [0.0, 116983.8, 45173.06, 0],
       [162597.7, 151377.59, 443898.53, 0],
       [93863.75, 127320.38, 249839.44, 1],
       [44069.95, 51283.14, 197029.42, 0],
       [77044.01, 99281.34, 140574.81, 2],
       [134615.46, 147198.87, 127716.82, 0],
       [67532.53, 105751.03, 304768.73, 1],
       [28663.76, 127056.21, 201126.82, 1],
       [78389.47, 153773.43, 299737.29, 2],
       [86419.7, 153514.11, 0.0, 2],
       [123334.88, 108679.17, 304981.62, 0],
       [38558.51, 82982.09, 174999.3, 0],
       [1315.46, 115816.21, 297114.46, 1],
       [144372.41, 118671.85, 383199.62, 2],
       [165349.2, 136897.8, 471784.1, 2],
       [0.0, 135426.92, 0.0, 0],
       [22177.74, 154806.14, 28334.72, 0]], dtype=object)
```

In [106]:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
print('Model has been trained successfully')
```

```
Model has been trained successfully
```

6.Model testing

In [107]:

```
y_pred=model.predict(x_test)
y_pred
```

Out[107]:

```
array([104055.1842384 , 132557.60289702, 133633.01284474,  72336.28081054,
       179658.27210893, 114689.63133397,  66514.82249033,  98461.69321326,
       114294.70487032, 169090.51127461,  96281.907934  ,  88108.30057881,
       110687.1172322 ,  90536.34203081, 127785.3793861 ])
```

In [108]:

```
testing_data_model_score = model.score(x_test, y_test)
print("Model Score/Performance on Testing data",testing_data_model_score)

training_data_model_score = model.score(x_train, y_train)
print("Model Score/Performance on Training data",training_data_model_score)
```

```
Model Score/Performance on Testing data 0.9355139722149947
Model Score/Performance on Training data 0.9515496105627431
```

In [109]:

```
df = pd.DataFrame(data={'Predicted value':y_pred.flatten(),'Actual Value':y_test.flatten()}
df                    #Comparing the predicted values and actual values
```

Out[109]:

	Predicted value	Actual Value
0	104055.184238	103282.38
1	132557.602897	144259.40
2	133633.012845	146121.95
3	72336.280811	77798.83
4	179658.272109	191050.39
5	114689.631334	105008.31
6	66514.822490	81229.06
7	98461.693213	97483.56
8	114294.704870	110352.25
9	169090.511275	166187.94
10	96281.907934	96778.92
11	88108.300579	96479.51
12	110687.117232	105733.54
13	90536.342031	96712.80
14	127785.379386	124266.90

Inference :

As we can see that the predicted value is close to the actual values i.e the one present in the testing set, Hence we can use this model for prediction. But first, we need to calculate how much is the error generated.

6.Model Evaluation

1. R2 score: R2 score – R squared score. It is one of the statistical approaches by whichwe can find the variance or the spread of the target and feature data.

In [110]:

```
from sklearn.metrics import r2_score

r2Score = r2_score(y_pred, y_test)
print("R2 score of model is :" ,r2Score*100)
```

```
R2 score of model is : 93.39448007716634
```

2. MSE: MSE – Mean Squared Error. By using this approach we can find that how much the regression best fit line is close to all the residual.

In [111]:

```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(y_pred, y_test)
print("Mean Squarred Error is :" ,mse*100)
```

```
Mean Squarred Error is : 6224496238.946447
```

3.RMSE: RMSE – Root Mean Squared Error

In [112]:

```
rmse = np.sqrt(mean_squared_error(y_pred, y_test))
print("Root Mean Squarred Error is : ",rmse*100)
```

```
Root Mean Squarred Error is :  788954.7666974608
```

==========

In [113]:

```
data1=startup_data.rename({'R&D Spend':'RDS','Administration':'ADMS','Marketing Spend':'MKT
data1
```

Out[113]:

	RDS	ADMS	MKTS	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
5	131876.90	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.60
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96
10	101913.08	110594.11	229160.95	Florida	146121.95
11	100671.96	91790.61	249744.55	California	144259.40
12	93863.75	127320.38	249839.44	Florida	141585.52
13	91992.39	135495.07	252664.93	California	134307.35
14	119943.24	156547.42	256512.92	Florida	132602.65
15	114523.61	122616.84	261776.23	New York	129917.04
16	78013.11	121597.55	264346.06	California	126992.93
17	94657.16	145077.58	282574.31	New York	125370.37
18	91749.16	114175.79	294919.57	Florida	124266.90
19	86419.70	153514.11	0.00	New York	122776.86
20	76253.86	113867.30	298664.47	California	118474.03
21	78389.47	153773.43	299737.29	New York	111313.02
22	73994.56	122782.75	303319.26	Florida	110352.25
23	67532.53	105751.03	304768.73	Florida	108733.99
24	77044.01	99281.34	140574.81	New York	108552.04
25	64664.71	139553.16	137962.62	California	107404.34
26	75328.87	144135.98	134050.07	Florida	105733.54
27	72107.60	127864.55	353183.81	New York	105008.31
28	66051.52	182645.56	118148.20	Florida	103282.38
29	65605.48	153032.06	107138.38	New York	101004.64
30	61994.48	115641.28	91131.24	Florida	99937.59
31	61136.38	152701.92	88218.23	New York	97483.56
32	63408.86	129219.61	46085.25	California	97427.84
33	55493.95	103057.49	214634.81	Florida	96778.92

	RDS	ADMS	MKTS	State	Profit
34	46426.07	157693.92	210797.67	California	96712.80
35	46014.02	85047.44	205517.64	New York	96479.51
36	28663.76	127056.21	201126.82	Florida	90708.19
37	44069.95	51283.14	197029.42	California	89949.14
38	20229.59	65947.93	185265.10	New York	81229.06
39	38558.51	82982.09	174999.30	California	81005.76
40	28754.33	118546.05	172795.67	California	78239.91
41	27892.92	84710.77	164470.71	Florida	77798.83
42	23640.93	96189.63	148001.11	California	71498.49
43	15505.73	127382.30	35534.17	New York	69758.98
44	22177.74	154806.14	28334.72	California	65200.33
45	1000.23	124153.04	1903.93	New York	64926.08
46	1315.46	115816.21	297114.46	Florida	49490.75
47	0.00	135426.92	0.00	California	42559.73
48	542.05	51743.15	0.00	New York	35673.41
49	0.00	116983.80	45173.06	California	14681.40

In [137]:

```
#Building SLR and MLR models for insignificant variavles'ADMS' & 'MKTS' &
#Also finding tvalues and pvalues

slr_ADMS=smf.ols('Profit~ADMS',data=data1).fit()   #SIMPLE LINEAR REGRESSION
slr_ADMS.tvalues,slr_ADMS.pvalues
```

Out[137]:

```
(Intercept    3.040044
 ADMS         1.419493
 dtype: float64,
 Intercept    0.003824
 ADMS         0.162217
 dtype: float64)
```

In [171]:

```
slr_MKTS=smf.ols('Profit~MKTS',data=data1).fit()      #SIMPLE LINEAR REGRESSION
slr_MKTS.tvalues,slr_MKTS.pvalues
```

Out[171]:

```
(Intercept    7.808356
 MKTS         7.802657
 dtype: float64,
 Intercept    4.294735e-10
 MKTS         4.381073e-10
 dtype: float64)
```

In [172]:

```
mlr_ADMS_MKTS=smf.ols('Profit~ADMS+MKTS',data=data1).fit()
mlr_ADMS_MKTS.tvalues,mlr_ADMS_MKTS.pvalues
```

Out[172]:

```
(Intercept    1.142741
 ADMS         2.467779
 MKTS         8.281039
 dtype: float64,
 Intercept    2.589341e-01
 ADMS         1.729198e-02
 MKTS         9.727245e-11
 dtype: float64)
```

7.Model Validation

Two Techniques: 1. Collinearity Check & 2. Residual Analysis

#1) Collinearity Problem Check

#Calculate VIF = 1/(1-Rsquare) for all independent variables

In [173]:

```
rsq_RDS=smf.ols('RDS~ADMS+MKTS',data=data1).fit().rsquared
vif_RDS=1/(1-rsq_RDS)

rsq_ADMS=smf.ols('ADMS~RDS+MKTS',data=data1).fit().rsquared
vif_ADMS=1/(1-rsq_ADMS)

rsq_MKTS=smf.ols('MKTS~RDS+ADMS',data=data1).fit().rsquared
vif_MKTS=1/(1-rsq_MKTS)

# putting in a data frame format
d1={'Variables':['RDS','ADMS','MKTS'],'VIF':[vif_RDS,vif_ADMS,vif_MKTS]}
pd.DataFrame(d1)
```

Out[173]:

	Variables	VIF
0	RDS	2.468903
1	ADMS	1.175091
2	MKTS	2.326773

 # None variable has VIF>20, No Collinearity, so consider all varaibles in Regression equation

In [141]:

```
#2) Residual Analysis

# Test for Normality of Residuals (Q-Q Plot) using residual model (model.resid)
```

In [142]:

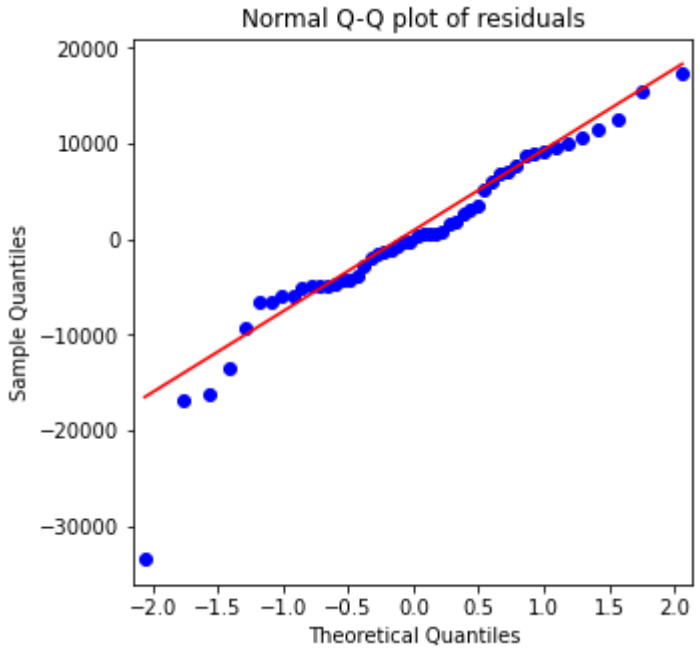
```
import statsmodels.api as sm
```

In [147]:

```
model=smf.ols("Profit~RDS+ADMS+MKTS",data=data1).fit()  #model building
```

In [148]:

```
sm.qqplot(model.resid,line='q')
plt.title('Normal Q-Q plot of residuals')
plt.show()
```



In [149]:

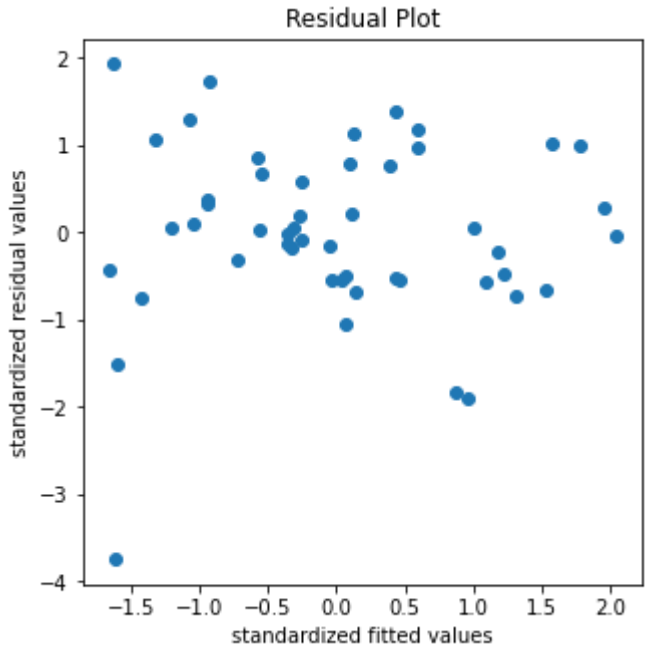
```
list(np.where(model.resid<-30000))
```

Out[149]:

```
[array([49], dtype=int64)]
```

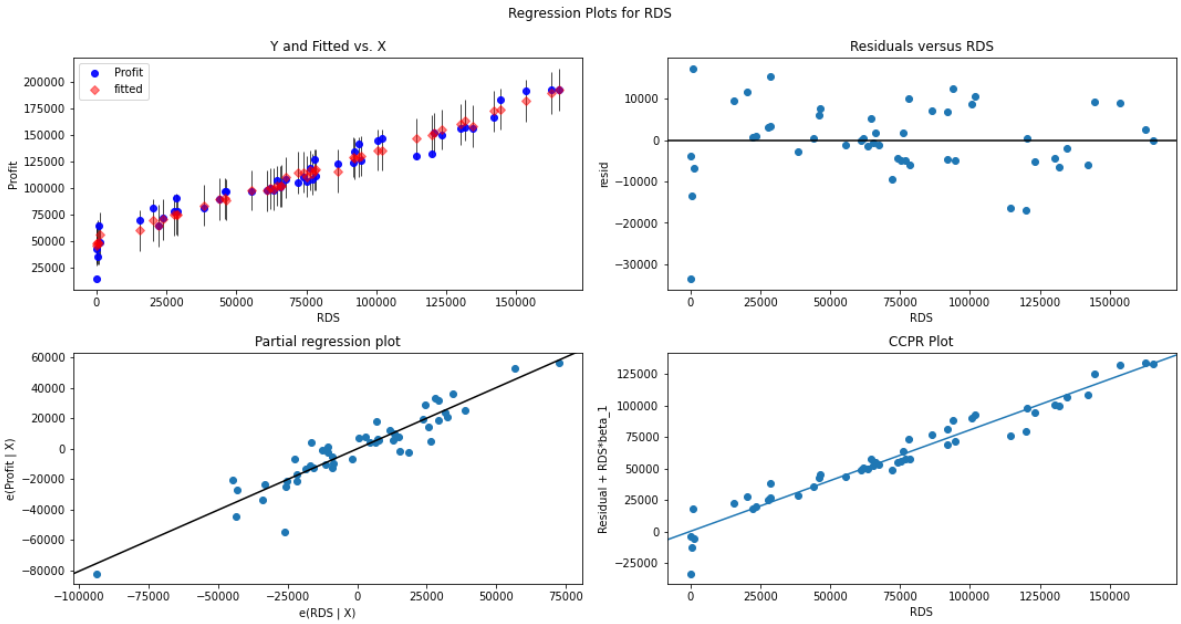
In [150]:

```
# Test for Homoscedasticity or Heteroscedasticity (plotting model's standardized fitted val
def standard_values(vals) : return (vals-vals.mean())/vals.std() # User defined z = (x - mu
plt.scatter(standard_values(model.fittedvalues),standard_values(model.resid))
plt.title('Residual Plot')
plt.xlabel('standardized fitted values')
plt.ylabel('standardized residual values')
plt.show()
```



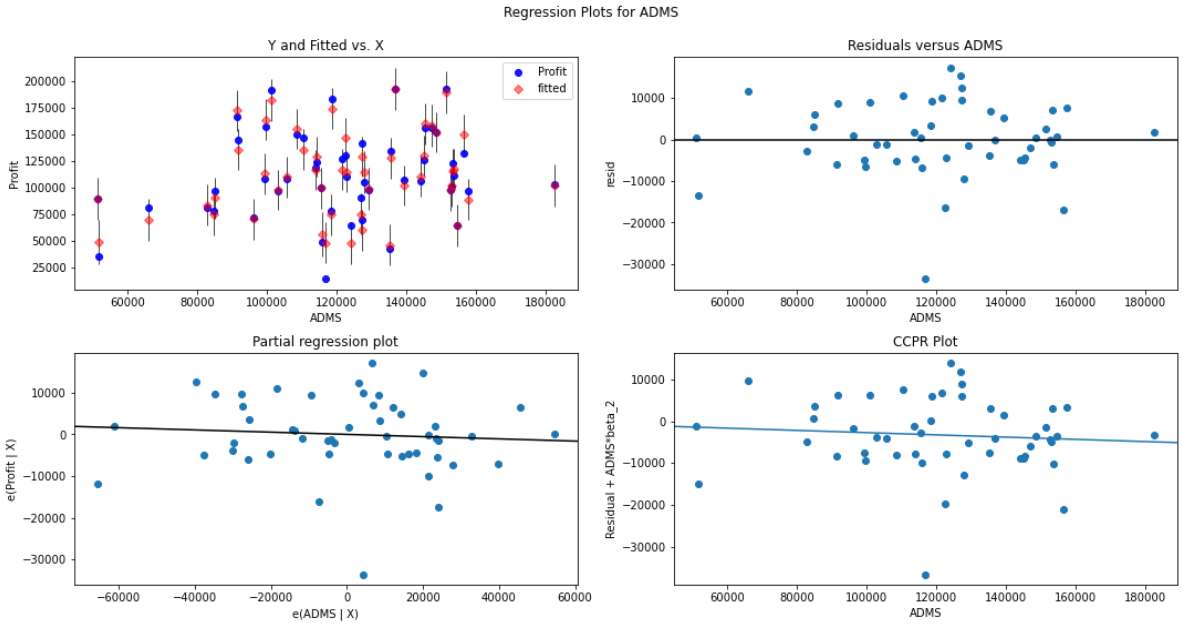
In [151]:

```
# Test for errors or Residuals Vs Regressors or independent 'x' variables or predictors
# using Residual Regression Plots code graphics.plot_regress_exog(model,'x',fig) # exog = x
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'RDS',fig=fig)
plt.show()
```



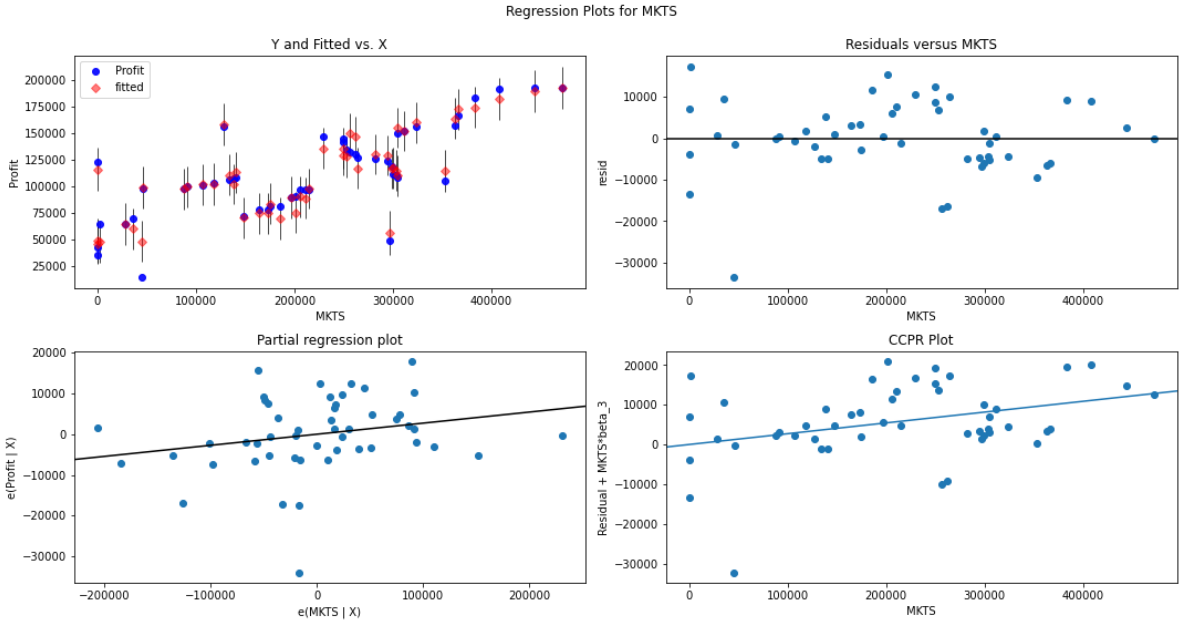
In [152]:

```
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'ADMS',fig=fig)
plt.show()
```



In [153]:

```
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'MKTS',fig=fig)
plt.show()
```



In [154]:

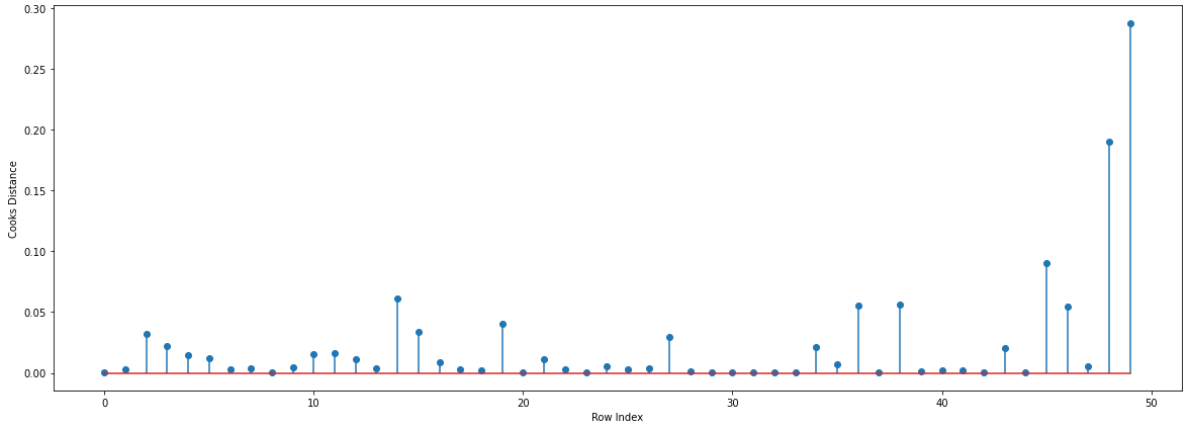
```
# Model Deletion Diagnostics (checking Outliers or Influencers)
# Two Techniques : 1. Cook's Distance & 2. Leverage value
# 1. Cook's Distance: If Cook's distance > 1, then it's an outlier
# Get influencers using cook's distance
(c,_)=model.get_influence().cooks_distance
c
```

Out[154]:

```
array([3.21825244e-05, 3.27591036e-03, 3.23842699e-02, 2.17206555e-02,
       1.44833032e-02, 1.17158463e-02, 2.91766303e-03, 3.56513444e-03,
       4.04303948e-05, 4.86758017e-03, 1.51064757e-02, 1.63564959e-02,
       1.15516625e-02, 4.01422811e-03, 6.12934253e-02, 3.40013448e-02,
       8.33556413e-03, 3.30534399e-03, 2.16819303e-03, 4.07440577e-02,
       4.25137222e-04, 1.09844352e-02, 2.91768000e-03, 2.76030254e-04,
       5.04643588e-03, 3.00074623e-03, 3.41957068e-03, 2.98396413e-02,
       1.31590664e-03, 1.25992620e-04, 4.18505125e-05, 9.27434786e-06,
       7.08656521e-04, 1.28122674e-04, 2.09815032e-02, 6.69508674e-03,
       5.55314705e-02, 6.55050578e-05, 5.61547311e-02, 1.54279607e-03,
       1.84850929e-03, 1.97578066e-03, 1.36089280e-04, 2.05553171e-02,
       1.23156041e-04, 9.03234206e-02, 5.45303387e-02, 5.33885616e-03,
       1.90527441e-01, 2.88082293e-01])
```

In [155]:

```
# Plot the influencers using the stem plot
fig=plt.figure(figsize=(20,7))
plt.stem(np.arange(len(data1)),np.round(c,5))
plt.xlabel('Row Index')
plt.ylabel('Cooks Distance')
plt.show()
```



In [156]:

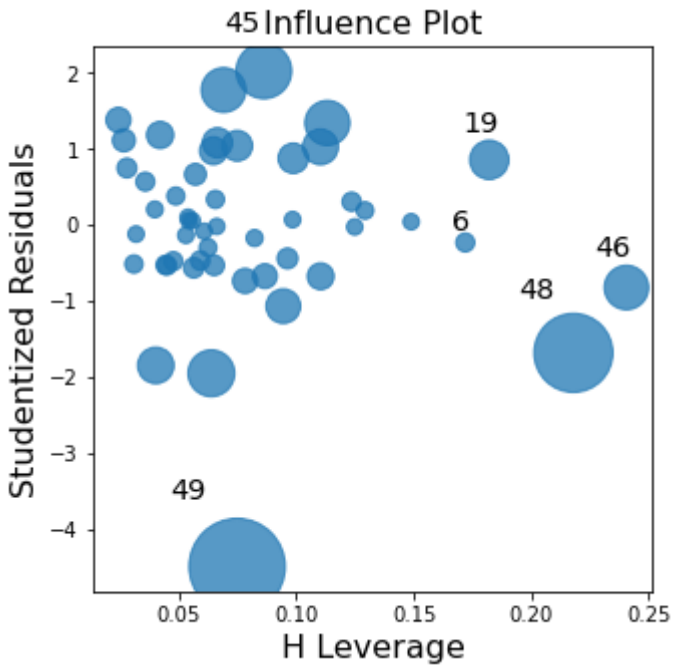
```
# Index and value of influencer where C>0.5
np.argmax(c) , np.max(c)
```

Out[156]:

```
(49, 0.2880822927543263)
```

In [157]:

```
# 2. Leverage Value using High Influence Points : Points beyond Leverage_cutoff value are i
influence_plot(model)
plt.show()
```



In [158]:

```
# Leverage Cuttoff Value = 3*(k+1)/n ; k = no.of features/columns & n = no. of datapoints
k=data1.shape[1]
n=data1.shape[0]
leverage_cutoff = (3*(k+1))/n
leverage_cutoff
```

Out[158]:

```
0.36
```

In [159]:

```
data1[data1.index.isin([49])]
```

Out[159]:

	RDS	ADMS	MKTS	State	Profit
49	0.0	116983.8	45173.06	California	14681.4

In [160]:

```
# Improving the Model
# Discard the data points which are influencers and reassign the row number (reset_index(dr
data2=data1.drop(data1.index[[49]],axis=0).reset_index(drop=True)
data2
```

Out[160]:

	RDS	ADMS	MKTS	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
5	131876.90	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.60
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96
10	101913.08	110594.11	229160.95	Florida	146121.95
11	100671.96	91790.61	249744.55	California	144259.40
12	93863.75	127320.38	249839.44	Florida	141585.52
13	91992.39	135495.07	252664.93	California	134307.35
14	119943.24	156547.42	256512.92	Florida	132602.65
15	114523.61	122616.84	261776.23	New York	129917.04
16	78013.11	121597.55	264346.06	California	126992.93
17	94657.16	145077.58	282574.31	New York	125370.37
18	91749.16	114175.79	294919.57	Florida	124266.90
19	86419.70	153514.11	0.00	New York	122776.86
20	76253.86	113867.30	298664.47	California	118474.03
21	78389.47	153773.43	299737.29	New York	111313.02
22	73994.56	122782.75	303319.26	Florida	110352.25
23	67532.53	105751.03	304768.73	Florida	108733.99
24	77044.01	99281.34	140574.81	New York	108552.04
25	64664.71	139553.16	137962.62	California	107404.34
26	75328.87	144135.98	134050.07	Florida	105733.54
27	72107.60	127864.55	353183.81	New York	105008.31
28	66051.52	182645.56	118148.20	Florida	103282.38
29	65605.48	153032.06	107138.38	New York	101004.64
30	61994.48	115641.28	91131.24	Florida	99937.59
31	61136.38	152701.92	88218.23	New York	97483.56
32	63408.86	129219.61	46085.25	California	97427.84

	RDS	ADMS	MKTS	State	Profit
33	55493.95	103057.49	214634.81	Florida	96778.92
34	46426.07	157693.92	210797.67	California	96712.80
35	46014.02	85047.44	205517.64	New York	96479.51
36	28663.76	127056.21	201126.82	Florida	90708.19
37	44069.95	51283.14	197029.42	California	89949.14
38	20229.59	65947.93	185265.10	New York	81229.06
39	38558.51	82982.09	174999.30	California	81005.76
40	28754.33	118546.05	172795.67	California	78239.91
41	27892.92	84710.77	164470.71	Florida	77798.83
42	23640.93	96189.63	148001.11	California	71498.49
43	15505.73	127382.30	35534.17	New York	69758.98
44	22177.74	154806.14	28334.72	California	65200.33
45	1000.23	124153.04	1903.93	New York	64926.08
46	1315.46	115816.21	297114.46	Florida	49490.75
47	0.00	135426.92	0.00	California	42559.73
48	542.05	51743.15	0.00	New York	35673.41

In [161]:

```
# Model Deletion Diagnostics and Final Model
while np.max(c)>0.5 :
 model=smf.ols("Profit~RDS+ADMS+MKTS",data=data2).fit()
 (c,_)=model.get_influence().cooks_distance
 c
 np.argmax(c) , np.max(c)
 data2=data2.drop(data2.index[[np.argmax(c)]],axis=0).reset_index(drop=True)
 data2
else:
 final_model=smf.ols("Profit~RDS+ADMS+MKTS",data=data2).fit()
 final_model.rsquared , final_model.aic
 print("Thus model accuracy is improved to",final_model.rsquared)
```

```
Thus model accuracy is improved to 0.9613162435129847
```

In [165]:

```
final_model.rsquared
```

Out[165]:

```
0.9613162435129847
```

In [166]:

```
data2
```

30	61994.48	115641.28	91131.24	Florida	99937.59
31	61136.38	152701.92	88218.23	New York	97483.56
32	63408.86	129219.61	46085.25	California	97427.84
33	55493.95	103057.49	214634.81	Florida	96778.92
34	46426.07	157693.92	210797.67	California	96712.80
35	46014.02	85047.44	205517.64	New York	96479.51
36	28663.76	127056.21	201126.82	Florida	90708.19
37	44069.95	51283.14	197029.42	California	89949.14
38	20229.59	65947.93	185265.10	New York	81229.06
39	38558.51	82982.09	174999.30	California	81005.76
40	28754.33	118546.05	172795.67	California	78239.91
41	27892.92	84710.77	164470.71	Florida	77798.83

In [167]:

```
# Model Predictions
# say New data for prediction is
new_data=pd.DataFrame({'RDS':70000,"ADMS":90000,"MKTS":140000},index=[0])
new_data
```

Out[167]:

	RDS	ADMS	MKTS
0	70000	90000	140000

In [168]:

```
# Manual Prediction of Price
final_model.predict(new_data)
```

Out[168]:

```
0    108727.154753
dtype: float64
```

In [169]:

```
# Automatic Prediction of Price with 90.02% accurcy
pred_y=final_model.predict(data2)
pred_y
```

Out[169]:

```
0     190716.676999
1     187537.122227
2     180575.526396
3     172461.144642
4     170863.486721
5     162582.583177
6     157741.338633
7     159347.735318
8     151328.826941
9     154236.846778
10    135507.792682
11    135472.855621
12    129355.599449
13    127780.129139
14    149295.404796
15    145937.941975
16    117437.627921
17    130408.626295
```

In [170]:

```
# table containing R^2 value for each prepared model
d2={'Prep_Models':['Model','Final_Model'],'Rsquared':[model.rsquared,final_model.rsquared]}
table=pd.DataFrame(d2)
table
```

Out[170]:

	Prep_Models	Rsquared
0	Model	0.950746
1	Final_Model	0.961316