In [28]:

```
import numpy as np
import pandas as pd
```

In [29]:

```
dataset=pd.read_csv(r"D:\ML_Course\Works_on_python\Multi_Linear_Regression\50_Startups.csv")
```

In [30]:

```
dataset.head()
```

Out[30]:

	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

In [31]:

```
dataset["State"].unique()
```

Out[31]:

```
array(['New York', 'California', 'Florida'], dtype=object)
```

In [32]:

```
dataset.isnull().any()
```

Out[32]:

```
R&D Spend          False
Administration     False
Marketing Spend    False
State              False
Profit             False
dtype: bool
```

In [33]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
dataset["State"]=le.fit_transform(dataset["State"])
```

In [34]:

```
dataset.head()
```

Out[34]:

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

In [35]:

```
x=dataset.iloc[:,0:4].values
y=dataset.iloc[:,4:5].values
```

In [36]:

```
x.shape
```

Out[36]:

```
(50, 4)
```

In [37]:

```
y.shape
```

Out[37]:

```
(50, 1)
```

In [38]:

```
from sklearn.preprocessing import OneHotEncoder
one=OneHotEncoder()
z=one.fit_transform(x[:,3:4]).toarray()
x=np.delete(x,3,axis=1)
x=np.concatenate((z,x),axis=1)
```

```
C:\Users\anikp\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:368: FutureWarning: The handling of integer data
will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the future
they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneH
otEncoder directly.
  warnings.warn(msg, FutureWarning)
```

```
In [39]: x.shape

Out[39]: (50, 6)
```

```
In [40]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

```
In [41]: x_test.shape

Out[41]: (10, 6)
```

```
In [42]: x_train.shape

Out[42]: (40, 6)
```

```
In [43]: y_train.shape

Out[43]: (40, 1)
```

```
In [44]: y_test.shape

Out[44]: (10, 1)
```

```
In [45]: from sklearn.linear_model import LinearRegression
         mlr=LinearRegression()
         mlr.fit(x_train,y_train)

Out[45]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                  normalize=False)
```

```
In [46]: y_pred=mlr.predict(x_test)
```

In [47]:
```
x_test
```

Out[47]:
```
array([[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 6.6051520e+04,
        1.8264556e+05, 1.1814820e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.0067196e+05,
        9.1790610e+04, 2.4974455e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.0191308e+05,
        1.1059411e+05, 2.2916095e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 2.7892920e+04,
        8.4710770e+04, 1.6447071e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.5344151e+05,
        1.0114555e+05, 4.0793454e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 7.2107600e+04,
        1.2786455e+05, 3.5318381e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 2.0229590e+04,
        6.5947930e+04, 1.8526510e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 6.1136380e+04,
        1.5270192e+05, 8.8218230e+04],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 7.3994560e+04,
        1.2278275e+05, 3.0331926e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.4210734e+05,
        9.1391770e+04, 3.6616842e+05]])
```

In [48]:
```
y_test
```

Out[48]:
```
array([[103282.38],
       [144259.4 ],
       [146121.95],
       [ 77798.83],
       [191050.39],
       [105008.31],
       [ 81229.06],
       [ 97483.56],
       [110352.25],
       [166187.94]])
```

```
In [49]: y_pred
```

```
Out[49]: array([[103015.20159794],
                [132582.27760817],
                [132447.73845174],
                [ 71976.09851258],
                [178537.48221058],
                [116161.24230168],
                [ 67851.69209676],
                [ 98791.73374684],
                [113969.43533015],
                [167921.06569553]])
```

```
In [50]: from sklearn.metrics import r2_score
         accuracy=r2_score(y_test,y_pred)
```

```
In [51]: accuracy
```

```
Out[51]: 0.9347068473282567
```

```
In [52]: #Random prediction, state col has been spliited into 3 cols and 1st enter it's value then rest
         y=mlr.predict([[0,0,1,12345,123456,123456]])#0,0,1 is for 1 is Florida and 6 cols so 6 values
```

```
In [53]: y
```

```
Out[53]: array([[61381.51739101]])
```

```
In [54]: y[0][0]
```

```
Out[54]: 61381.51739101054
```

Decision Tree

In [59]:
```
from sklearn.tree import DecisionTreeRegressor
dtr=DecisionTreeRegressor(random_state=0)
dtr.fit(x_train,y_train)
```

Out[59]:
```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=0, splitter='best')
```

In [80]:
```
y_dtr=dtr.predict(x_test)
```

In [81]:
```
y_dtr
```

Out[81]:
```
array([101004.64, 141585.52, 141585.52,  78239.91, 182901.99, 107404.34,
        69758.98,  97427.84, 108733.99, 182901.99])
```

In [82]:
```
y_test
```

Out[82]:
```
array([[103282.38],
       [144259.4 ],
       [146121.95],
       [ 77798.83],
       [191050.39],
       [105008.31],
       [ 81229.06],
       [ 97483.56],
       [110352.25],
       [166187.94]])
```

In [83]:
```
accuracy_dtr=r2_score(y_test,y_dtr)
```

In [84]:
```
accuracy_dtr # more accuracy
```

Out[84]:
```
0.9594341740623319
```

```
for draw decision tree
```

```
from sklearn import tree
tree.export_graphviz(dtr)
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(dtr, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

In [70]:

```
yd=dtr.predict([[1,0,0,12345,23456,6789]])
```

In [71]:

```
yd
```

Out[71]:

```
array([65200.33])
```

Random Forest

In [72]:

```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(n_estimators = 10,random_state = 0)#n_estimator is more then more accu or may be not
rfr.fit(x_train,y_train)
```

```
C:\Users\anikp\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was passed when a
1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[72]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False)
```

In [77]:

```
y_rfr=rdr.predict(x_test)
```

In [78]:

```
accuracy_rfr=r2_score(y_test,y_rfr)
```

```
In [79]: accuracy_rfr#omore accuracy

Out[79]: 0.9658739721928109

In [ ]:
```