Data preprocessing and exploration

In [32]:
```
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [33]:
```
dataset = pd.read_csv('50_Startups.csv')
dataset.head()
```

Out[33]:

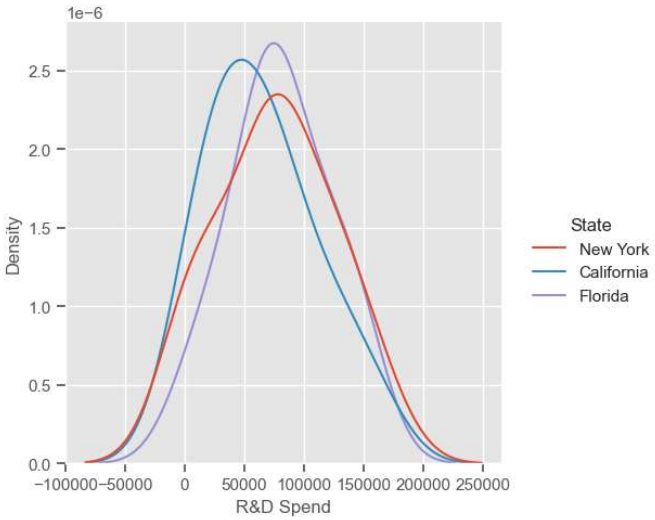
	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

In [34]:
```
dataset.shape
```

Out[34]:
```
(50, 5)
```

In [35]:
```
sns.displot(data=dataset, x='R&D Spend', hue='State', kind='kde')
```

Out[35]:
```
<seaborn.axisgrid.FacetGrid at 0x21421d1e710>
```

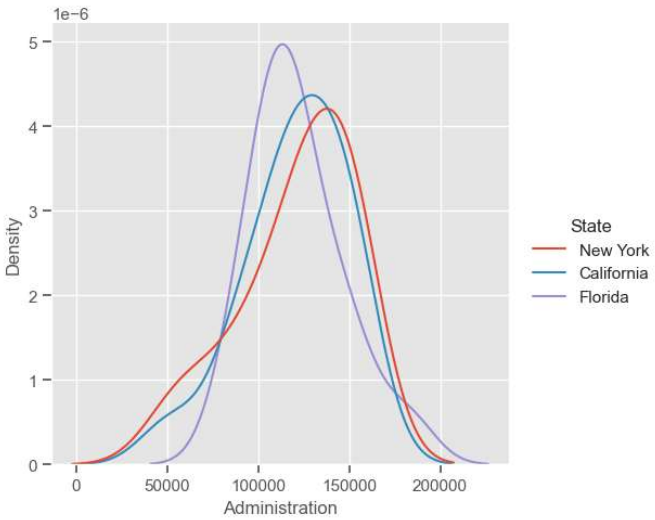


In [36]:
```
from scipy.stats import kurtosis, skew
print("skew : ",skew(dataset['R&D Spend']))
print("kurt : ",kurtosis(dataset['R&D Spend']))
```

```
skew :  0.15904052321503395
kurt :  -0.8050676431519039
```

In [37]:
```
sns.displot(data=dataset, x='Administration', hue='State', kind='kde')
```

Out[37]:
```
<seaborn.axisgrid.FacetGrid at 0x21421f3d090>
```

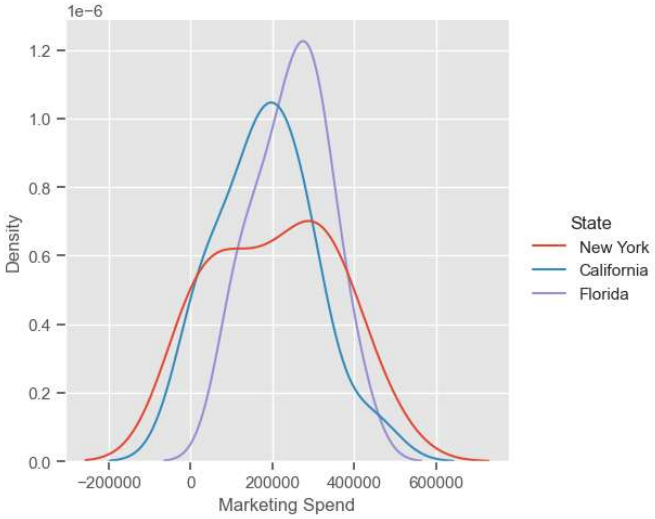


In [38]:
```
print("skew : ",skew(dataset['Administration']))
print("kurt : ",kurtosis(dataset['Administration']))
```

```
skew :  -0.4742300698920047
kurt :  0.08553840791984113
```

In [39]:
```
sns.displot(data=dataset, x='Marketing Spend', hue='State', kind='kde')
```

Out[39]:
```
<seaborn.axisgrid.FacetGrid at 0x21421f87ed0>
```

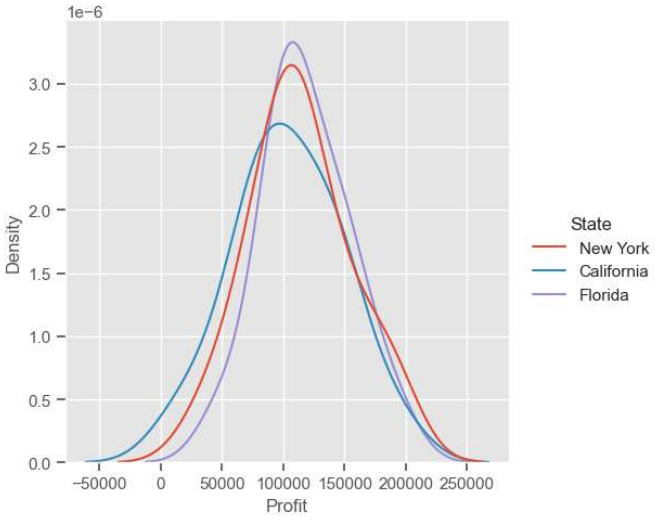


In [40]:
```
print("skew : ",skew(dataset['Marketing Spend']))
print("kurt : ",kurtosis(dataset['Marketing Spend']))
```

```
skew :  -0.04506631617666136
kurt :  -0.7240327111087308
```

In [41]:
```
sns.displot(data=dataset, x='Profit', hue='State', kind='kde')
```

Out[41]:
```
<seaborn.axisgrid.FacetGrid at 0x21421d1e490>
```



In [42]:
```
print("skew : ",skew(dataset['Profit']))
print("kurt : ",kurtosis(dataset['Profit']))
```

```
skew :  0.02258638356958943
kurt :  -0.17529637679752152
```

In [43]:
```
dataset.describe()
```

Out[43]:

	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 [44]:
```
dataset.loc[dataset['R&D Spend'] == 0.0]
```

Out[44]:

	R&D Spend	Administration	Marketing Spend	State	Profit
47	0.0	135426.92	0.00	California	42559.73
49	0.0	116983.80	45173.06	California	14681.40

Recall some basics statistical functions

$$mean := m = \frac{1}{n}\sum_{i=1}^{n} x_i,\ variance := var = \frac{1}{n}\sum_{i=1}^{n}(x_i - m)^2,\ ecart_type := std = \sqrt{var}$$

In [45]:
```
dataset.describe(include='object')
```

Out[45]:

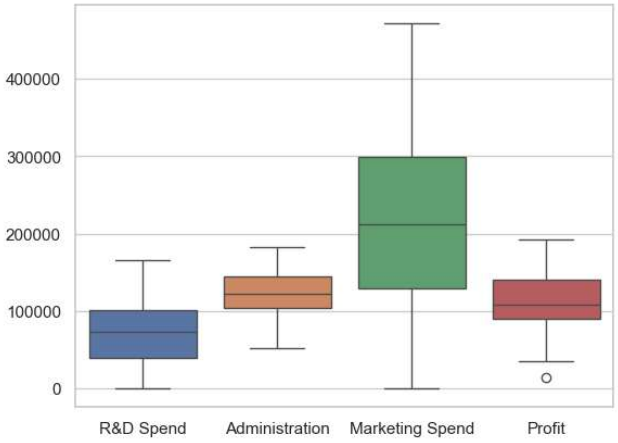
	State
count	50
unique	3
top	New York
freq	17

In [46]:
```
dataset.State.value_counts()
```

Out[46]:
```
State
New York      17
California    17
Florida       16
Name: count, dtype: int64
```

In [47]:
```
sns.set(style="whitegrid")
```

In [48]:
```
ax = sns.boxplot(data=dataset, orient="v")
```

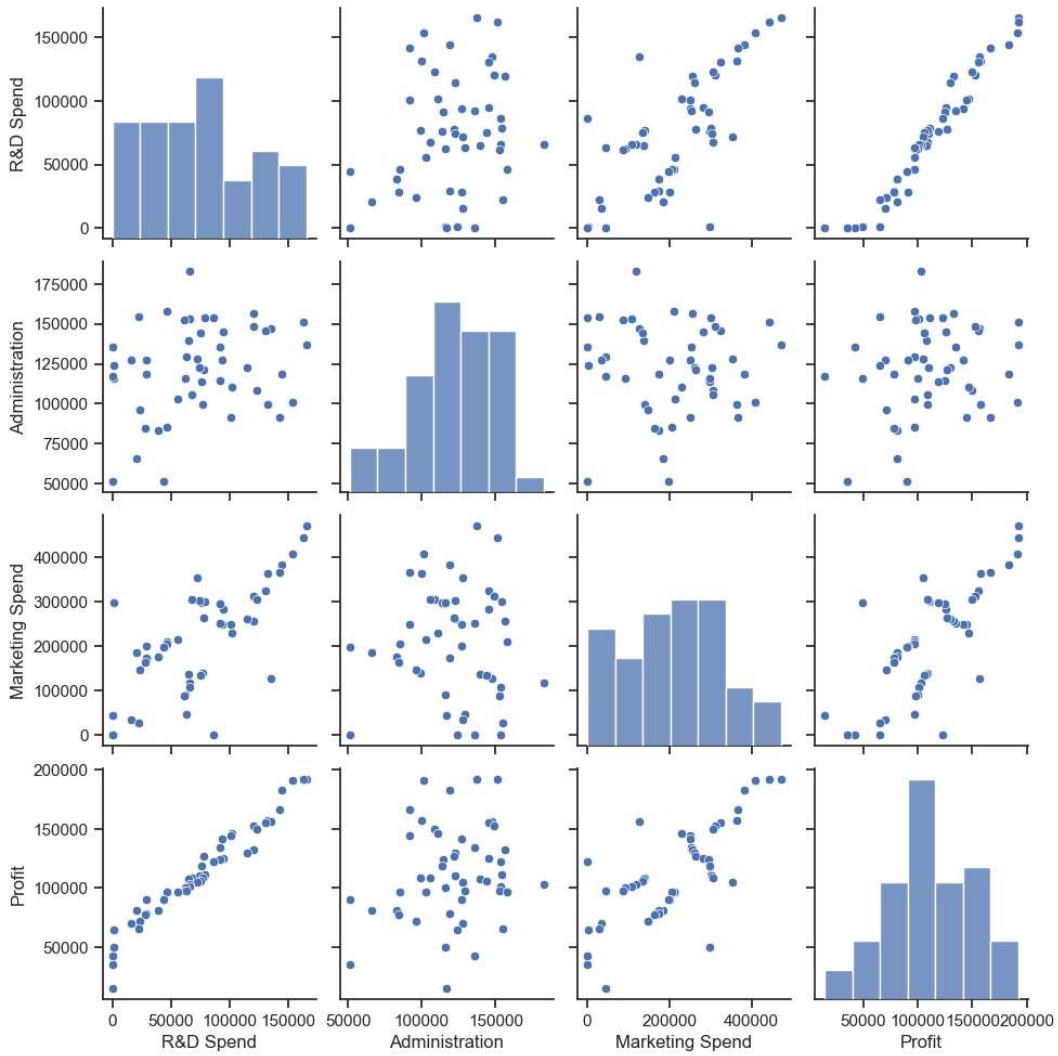


In [49]:
```
dataset.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   R&D Spend        50 non-null     float64
 1   Administration   50 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 [50]:
```
sns.set(style="ticks")
```

In [51]:
```
sns.pairplot(dataset)
```

Out[51]:
```
<seaborn.axisgrid.PairGrid at 0x214222c2d50>
```



In [ ]:

Recall the covariance and the Pearson correlation coefficient

$$cov(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - mean_i)\left(y_j - mean_j\right), \quad pearson := \rho_{x,y} = \frac{cov(x,y)}{std_x std_y}$$

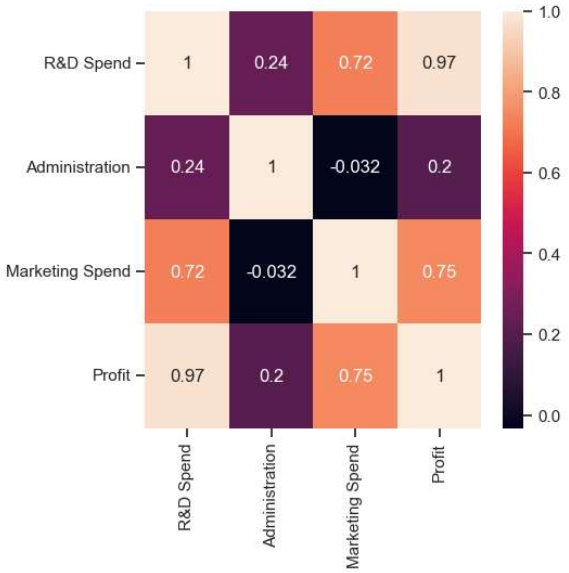
In [52]:
```
dataset.columns
```

Out[52]:
```
Index(['R&D Spend', 'Administration', 'Marketing Spend', 'State', 'Profit'], dtype='object')
```

In [53]:
```
correlations = dataset[['R&D Spend', 'Administration', 'Marketing Spend', 'Profit']].corr(method='pearson')
```

In [54]:
```
f, ax = plt.subplots(figsize = (5, 5))
sns.heatmap(correlations, annot = True)
```

Out[54]:
```
<Axes: >
```



In [55]:
```
X = dataset.iloc[:,0:4].values
y = dataset.iloc[:,4].values
```

In [56]:
```
X[0:10,:]
```

Out[56]:
```
array([[165349.2, 136897.8, 471784.1, 'New York'],
       [162597.7, 151377.59, 443898.53, 'California'],
       [153441.51, 101145.55, 407934.54, 'Florida'],
       [144372.41, 118671.85, 383199.62, 'New York'],
       [142107.34, 91391.77, 366168.42, 'Florida'],
       [131876.9, 99814.71, 362861.36, 'New York'],
       [134615.46, 147198.87, 127716.82, 'California'],
       [130298.13, 145530.06, 323876.68, 'Florida'],
       [120542.52, 148718.95, 311613.29, 'New York'],
       [123334.88, 108679.17, 304981.62, 'California']], dtype=object)
```

In [57]:
```
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
X[:,3] = labelencoder_X.fit_transform(X[:,3])
X[0:5,:]
```

Out[57]:
```
array([[165349.2, 136897.8, 471784.1, 2],
       [162597.7, 151377.59, 443898.53, 0],
       [153441.51, 101145.55, 407934.54, 1],
       [144372.41, 118671.85, 383199.62, 2],
       [142107.34, 91391.77, 366168.42, 1]], dtype=object)
```

In [58]:
```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
ct = ColumnTransformer([("State", OneHotEncoder(), [3])], remainder = 'passthrough')
X = ct.fit_transform(X)
X = X.astype('float64')
X.shape
```

Out[58]:
```
(50, 6)
```

In [59]:
```
X[0:5,:]
```

Out[59]:
```
array([[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.6534920e+05,
        1.3689780e+05, 4.7178410e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.6259770e+05,
        1.5137759e+05, 4.4389853e+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, 1.4437241e+05,
        1.1867185e+05, 3.8319962e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.4210734e+05,
        9.1391770e+04, 3.6616842e+05]])
```

In [60]:
```
# Avoid the trap of the Dummy variable
if X.shape[1]==6:
    X = X[:,1:]
X.shape
```

Out[60]:
```
(50, 5)
```

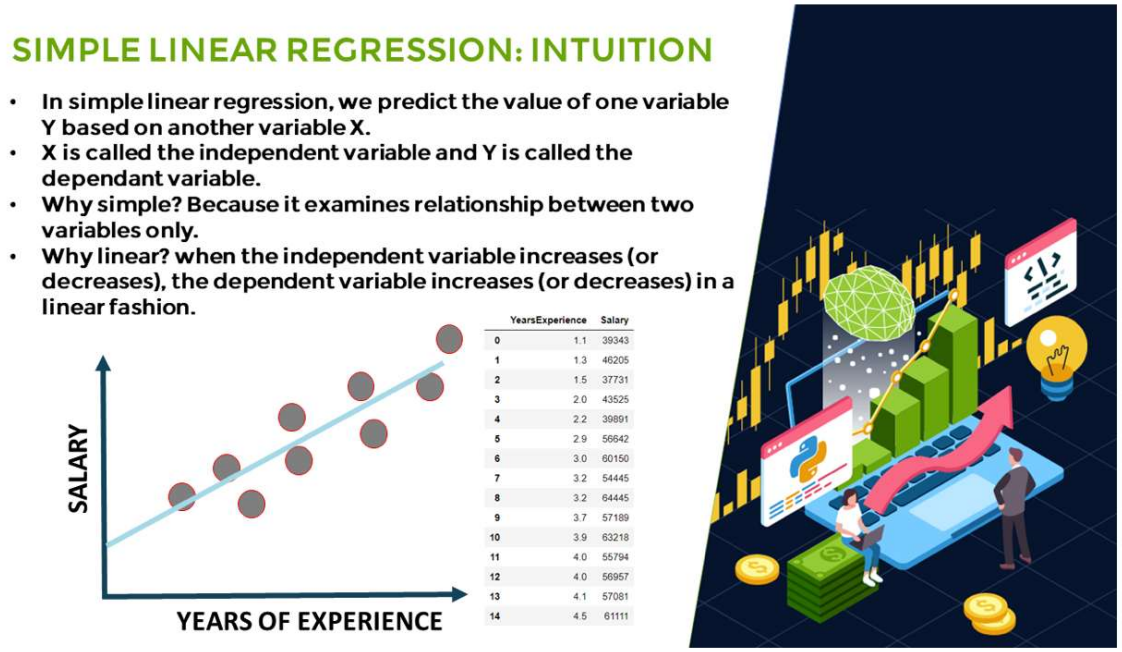
In [61]:
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)
X_train.shape
```

Out[61]:
```
(40, 5)
```

Data modeling

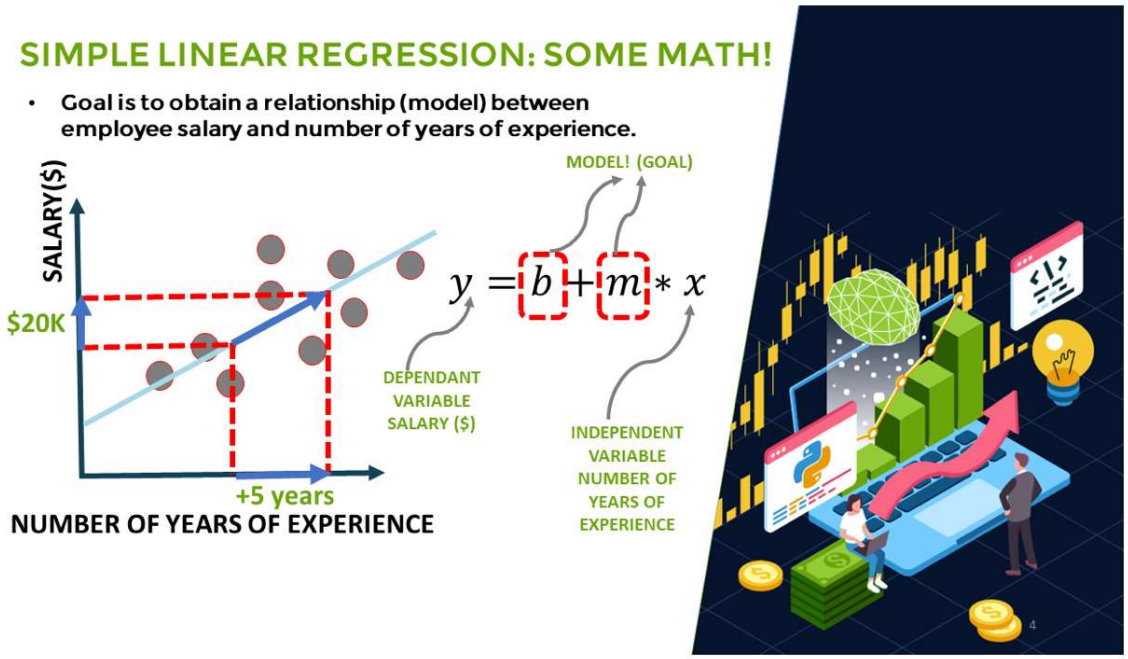
In [62]:
```
from IPython.display import display, Image
Image(filename='regression1.png')
```

Out[62]:



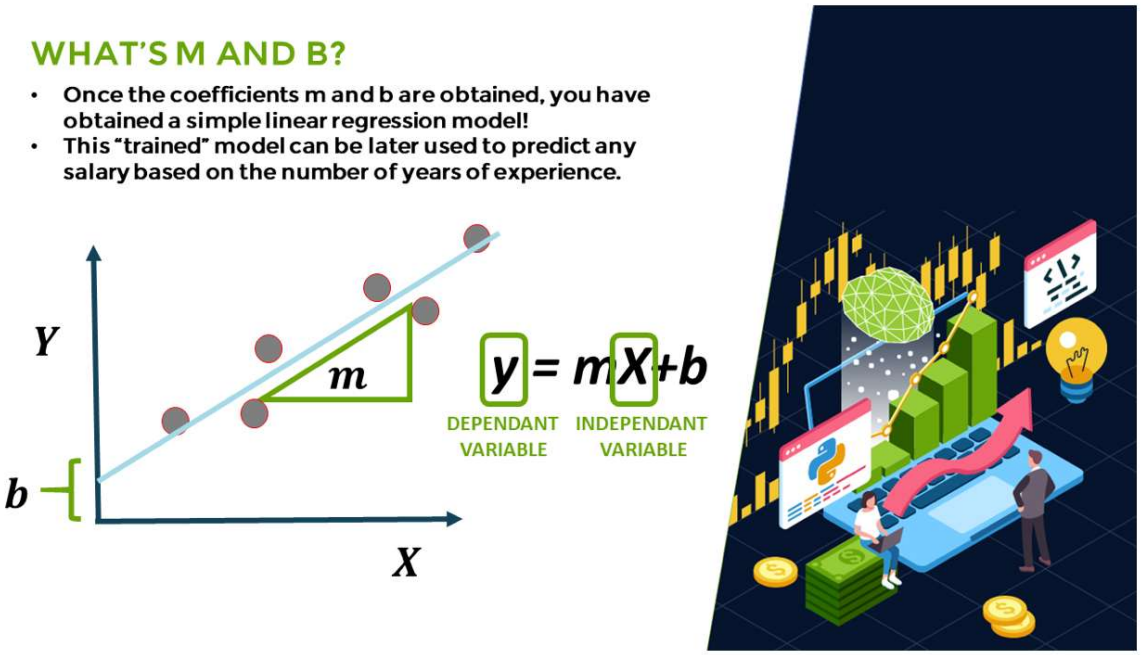
In [63]:
```
Image(filename='regression2.png')
```

Out[63]:



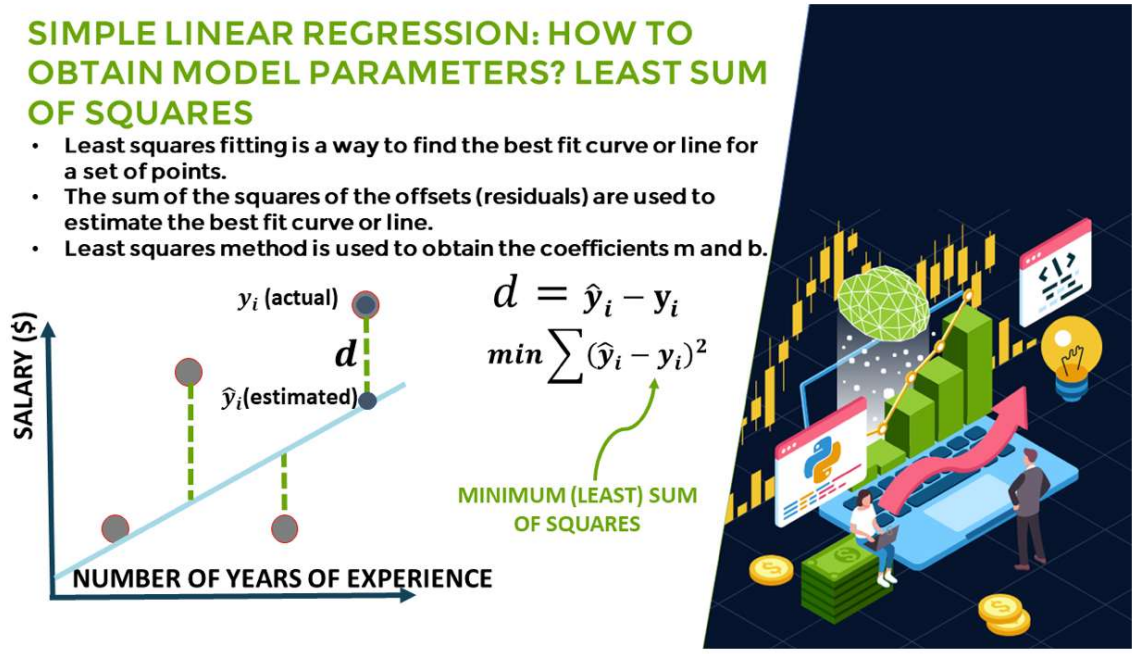
In [64]:
```
Image(filename='regression3.png')
```

Out[64]:



In [65]:
```
Image(filename='regression4.png')
```

Out[65]:



In [66]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,y_train)
```

Out[66]:

LinearRegression

▶ Parameters

In [69]:
```
print(f'b = {regressor.intercept_}')
print(f'm_i = {regressor.coef_}')
```

```
b = 42554.16761774421
m_i = [-9.59284160e+02  6.99369053e+02  7.73467193e-01  3.28845975e-02
  3.66100259e-02]
```

In [72]:
```
y_pred = regressor.predict(X_test)
```

In [73]:
```
for i in range(10):
    print([y_test[i],y_pred[i]])
```

```
[np.float64(103282.38), np.float64(103015.20159796975)]
[np.float64(144259.4), np.float64(132582.27760815)]
[np.float64(146121.95), np.float64(132447.73845174478)]
[np.float64(77798.83), np.float64(71976.09851258468)]
[np.float64(191050.39), np.float64(178537.48221053835)]
[np.float64(105008.31), np.float64(116161.24230164225)]
[np.float64(81229.06), np.float64(67851.69209675884)]
[np.float64(97483.56), np.float64(98791.73374688232)]
[np.float64(110352.25), np.float64(113969.43533012061)]
[np.float64(166187.94), np.float64(167921.0656954947)]
```

In [74]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

In [75]:
```
def rmse(targets, predictions):
    return np.sqrt(((predictions - targets) ** 2).mean())
```

In [76]:
```
print(f'MSE = {mean_squared_error(y_test, y_pred)}')
print(f'RMSE = {rmse(y_test, y_pred)}')
print(f'MAE = {mean_absolute_error(y_test, y_pred)}')
print(f'R2 = {r2_score(y_test,y_pred)}')
```

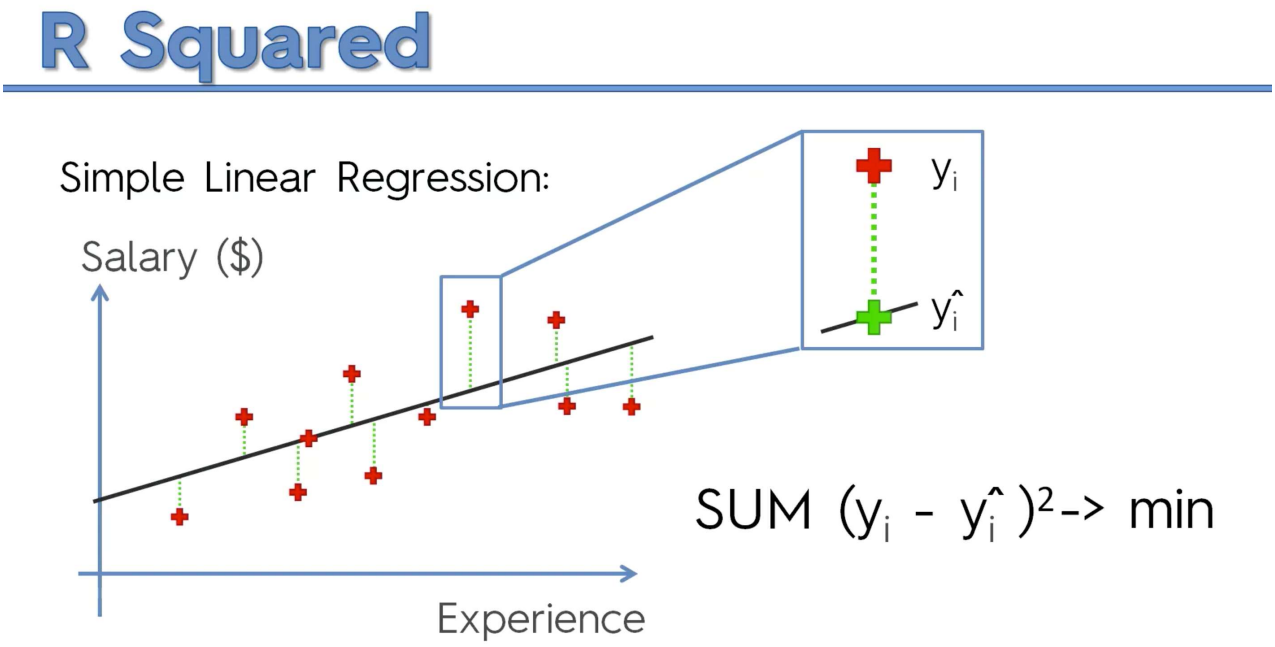
```
MSE = 83502864.03258388
RMSE = 9137.9901527953
MAE = 7514.293659639353
R2 = 0.9347068473282374
```

In [85]:
```
y.mean()
```

Out[85]:
```
np.float64(112012.63920000002)
```

In [77]:
```
Image(filename='RSq1.png')
```

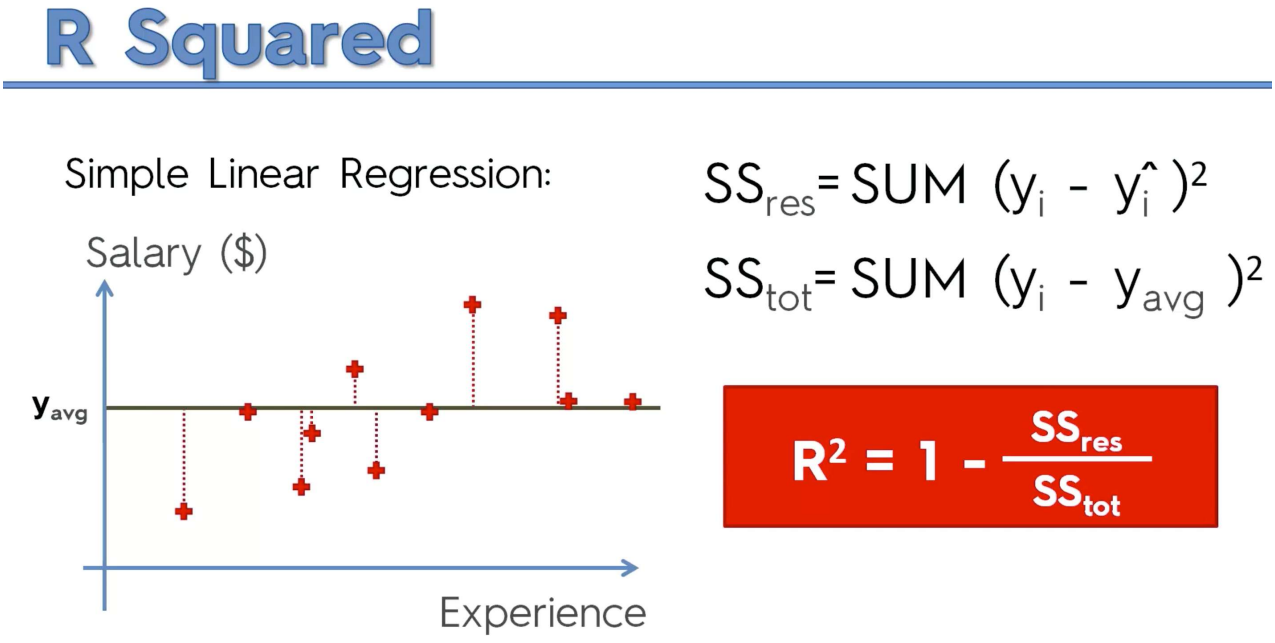
Out[77]:



residual sum of squares total sum of squares

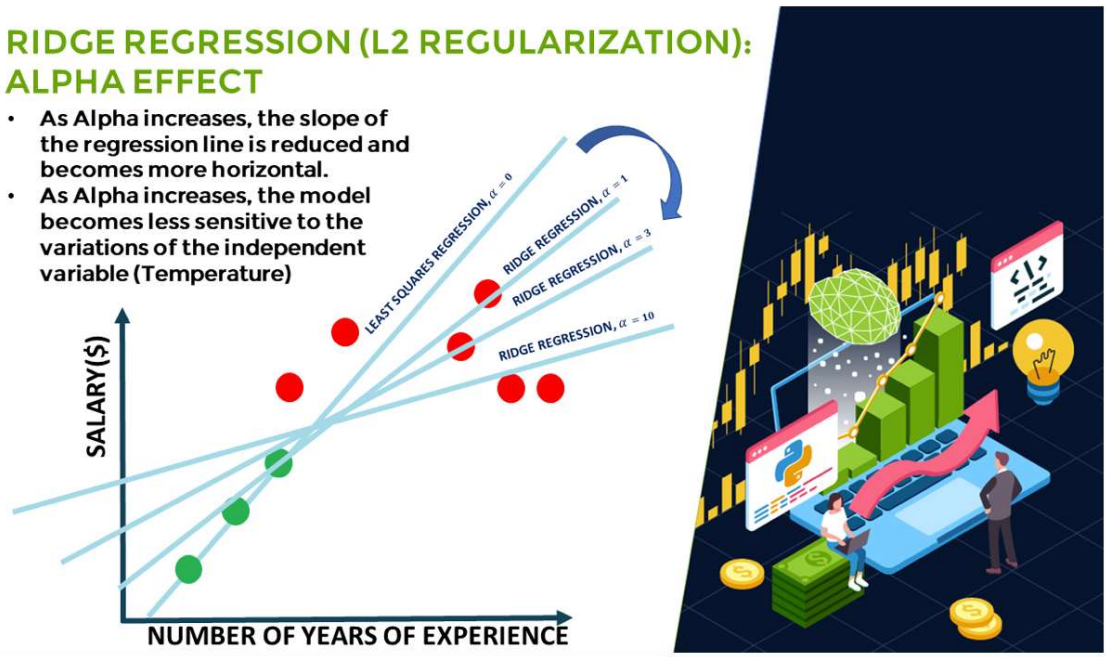
In [78]:
```
Image(filename='RSq2.png')
```

Out[78]:



In [79]:
```
Image(filename='regularization6.png')
```

Out[79]:



In [88]:
```
from sklearn.ensemble import RandomForestRegressor

ridge = RandomForestRegressor()
ridge.fit(X_train, y_train)
```

Out[88]:

RandomForestRegressor

▶ Parameters

In [89]:
```
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[89], line 1
----> 1 print(f'intercept  : {ridge.intercept_}')
      2 print(f'coef :  {ridge.coef_}')

AttributeError: 'RandomForestRegressor' object has no attribute 'intercept_'
```

In [90]:
```
y_pred_r = ridge.predict(X_test)
```

In [91]:
```
for i in range(10):
    print([y_test[i],y_pred_r[i]])
```

```
[np.float64(103282.38), np.float64(103820.33699999994)]
[np.float64(144259.4), np.float64(134681.93529999978)]
[np.float64(146121.95), np.float64(134694.17089999982)]
[np.float64(77798.83), np.float64(78244.16390000001)]
[np.float64(191050.39), np.float64(182943.11520000012)]
[np.float64(105008.31), np.float64(115881.13579999989)]
[np.float64(81229.06), np.float64(69835.7784000001)]
[np.float64(97483.56), np.float64(99052.65219999991)]
[np.float64(110352.25), np.float64(112651.73659999996)]
[np.float64(166187.94), np.float64(168355.00279999987)]
```

In [92]:
```
print(f'MSE = {mean_squared_error(y_test, y_pred_r)}')
print(f'RMSE = {rmse(y_test, y_pred_r)}')
print(f'MAE = {mean_absolute_error(y_test, y_pred_r)}')
print(f'R2 = {r2_score(y_test,y_pred_r)}')
```

```
MSE = 54900864.656606555
RMSE = 7409.5117691118185
MAE = 5839.755849999978
R2 = 0.9570715258768033
```

In [ ]:
```
MSE = 83502864.03258388
RMSE = 9137.9901527953
MAE = 7514.293659639353
R2 = 0.9347068473282374
```

In [ ]:
```
```