Supervised Machine Learning (Regression)

SALARY AND EXPERINCE DATA SET

IMPORTING LIBRARIES

In [2]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # MATLAB-like way of plotting
```

sklearn package for machine learning in python

In [3]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [23]:
```
df = pd.read_csv(r"C:\Users\owner\Downloads\salary_data (1).csv")
```

EDA

In [24]:
```
df.head()
```

Out[24]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

In [25]:
```
df.shape
```

Out[25]:
```
(30, 2)
```

In [26]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 612.0 bytes
```

In [27]:
```
df.columns
```

Out[27]:
```
Index(['YearsExperience', 'Salary'], dtype='object')
```

In [28]:
```
df.describe()
```

Out[28]:

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [29]:
```
df.corr()
```

Out[29]:

	YearsExperience	Salary
YearsExperience	1.000000	0.978242
Salary	0.978242	1.000000

In [32]:
```
df.corr(),'\n'
```

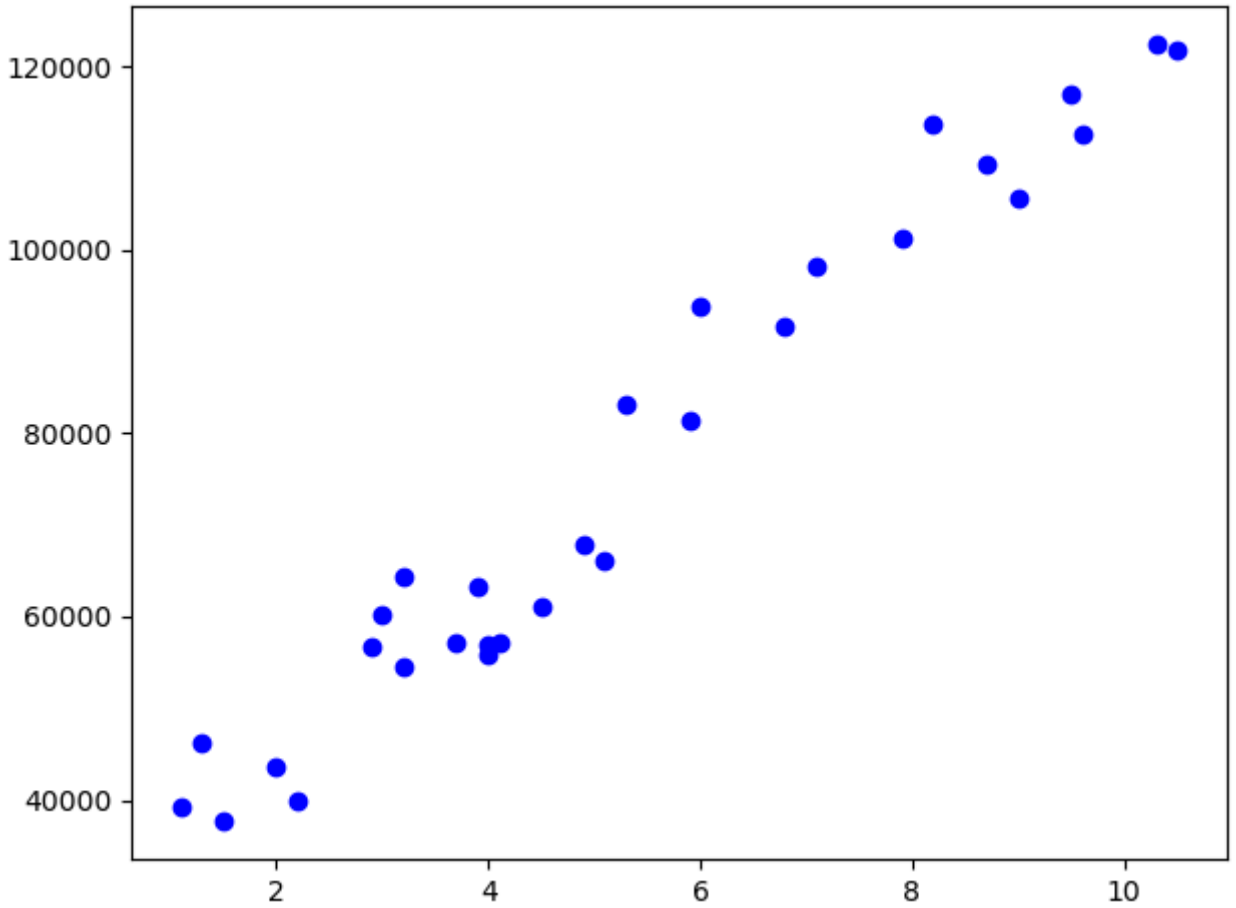
Out[32]:
```
(                 YearsExperience    Salary
 YearsExperience         1.000000  0.978242
 Salary                  0.978242  1.000000,
 '\n')
```

LINEAR REGRESSION

In [34]:
```
X = df.iloc[:, [0]].values # inputs YearsExperience
y = df.iloc[:, 1].values # outputs Salary
```

In [35]:
```
# visualise initial data set
fig1, ax1 = plt.subplots()
ax1.scatter(X, y, color = 'blue')
```

```
fig1.tight_layout()
fig1.savefig('LR_initial_plot.png')
```



SPLITTING DATASET INTO TRAIN AND TEST

In [36]:
```
# split the data into training and test sets:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3,
random_state = 0)
```

In [37]:
```
# fit the linear least-squares regression line to the training data:
regr = LinearRegression()
```

```
regr.fit(X_train, y_train)
```

Out[37]:

▾ LinearRegression

LinearRegression()

In [41]:

```
# visualise training set results
fig2, ax2 = plt.subplots()
ax2.scatter(X_train, y_train, color = 'red')
ax2.plot(X_train, regr.predict(X_train), color = 'blue')
ax2.set_title('Salary vs Experience (Train set)')
ax2.set_xlabel('Years of Experience')
ax2.set_ylabel('Salary')
fig2.tight_layout()
fig2.savefig('LR_train_plot.png')
```

In [39]:
```
# visualise test set results
fig3, ax3 = plt.subplots()
ax3.scatter(X_test, y_test, color = 'red')
ax3.plot(X_test, regr.predict(X_test), color = 'blue')
ax3.set_title('Salary vs Experience (Test set)')
ax3.set_xlabel('Years of Experience')
ax3.set_ylabel('Salary')
fig3.tight_layout()
fig3.savefig('LR_test_plot.png')
```

TESTING MODEL

In [42]:
```
# The coefficients
print('Coefficients: ', regr.coef_)

# The intercept
print('Intercept: ', regr.intercept_)

# The mean squared error
print('Mean squared error: %.8f'
% mean_squared_error(y_test, regr.predict(X_test)))
```

```
# The R^2 value:
print('Coefficient of determination: %.2f'
% r2_score(y_test, regr.predict(X_test)))
```

```
Coefficients:  [9345.94244312]
Intercept:  26816.19224403119
Mean squared error: 21026037.32951130
Coefficient of determination: 0.97
```

PREDICTION

In [43]:
```
print('Predict single value: ', regr.predict(np.array([[6]])))
```

```
Predict single value:  [82891.84690277]
```

In [44]:
```
print('Predict single value: ', regr.predict(np.array([[10]])))
```

```
Predict single value:  [120275.61667525]
```

In [45]:
```
print('Predict single value: ', regr.predict(np.array([[15]])))
```

```
Predict single value:  [167005.32889087]
```

In [ ]:
```

```