```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

```
data = pd.read_csv('Salary_Data.csv')
data.head(5)
```

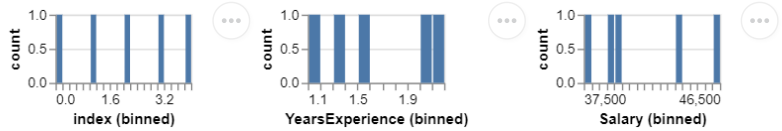
1 to 5 of 5 entries Filter

index	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

Show 25 per page

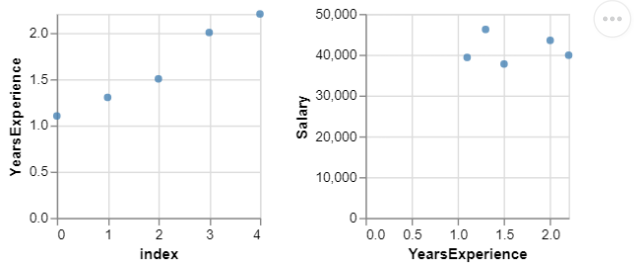
Like what you see? Visit the data table notebook to learn more about interactive tables.

Distributions



Values

2-d distributions




```
data.shape
```

```
(30, 2)
```

```
data.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: 608.0 bytes
```

```
data.describe()
```

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785

```
data.isnull().sum()
```

```
YearsExperience    0
Salary             0
dtype: int64
```

```
num_duplicates = data.duplicated().sum()
if num_duplicates > 0:
  print(f"The dataset contains {num_duplicates} duplicate values.")
  data = data.drop_duplicates()
  print("Dropped duplicates.")
  print("Number of Duplicate Values after dropping :",num_duplicates)
else:
  print("The dataset doesn't contain any duplicate values.")
```

```
The dataset doesn't contain any duplicate values.
```

```
 X = data.iloc[:,:-1] # Independent feature
X.head(5)
```

	YearsExperience
0	1.1
1	1.3
2	1.5
3	2.0
4	2.2

Preparing the data

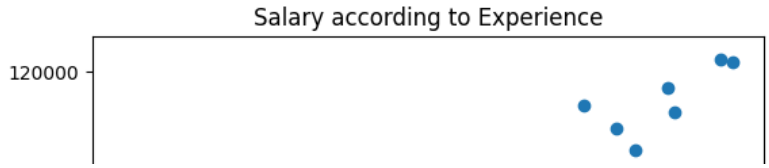
```
Y = data.iloc[:,-1] # Dependent feature
Y.head(5)
```

```
0    39343.0
1    46205.0
2    37731.0
3    43525.0
4    39891.0
Name: Salary, dtype: float64
```

Plotting the data to a look of the data distribution

```
 plt.scatter(X,Y)
plt.title("Salary according to Experience")
plt.xlabel("Salary")
plt.ylabel("Years of experience")
```

```
Text(0, 0.5, 'Years of experience')
```



Splitting the dataset into train and test

```
 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33,
random_state=51)
```

```
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(20, 1)
(10, 1)
(20,)
(10,)
```

Training the model

```
 linear = LinearRegression()
```

```
linear.fit(X_train, Y_train)
```

```
▾ LinearRegression
LinearRegression()
```

Checking intercept and coeficient(slope)

```
linear.coef_
```

```
array([9523.14578831])
```

```
linear.intercept_
```

```
24006.03576146961
```

Testing the model

```
Y_pred = linear.predict(X_test)
```

Double-click (or enter) to edit

```
 Y_pred
```

```
array([106857.40411979,  54480.10228407,  38290.75444394, 102095.83122563,
        54480.10228407, 115428.23532927,  70669.4501242 ,  80192.59591251,
        36386.12528628,  81144.91049134])
```

```
 Y_test
```

```
24    109431.0
8      64445.0
2      37731.0
23    113812.0
7      54445.0
27    112635.0
15     67938.0
18     81363.0
```

```
1      46205.0
19     93940.0
Name: Salary, dtype: float64
```

```
score = r2_score(Y_test, Y_pred)
print(f"Score: {score *100}")
```

```
Score: 92.78148083974355
```

Here is Plotting the graph

```
# Plotting the scatter plot of actual data points
plt.scatter(X_test, Y_test, color='blue', label='Actual')
plt.plot(X_test, Y_pred, color='red', linewidth=2, label='Predicted')
plt.title("Salary Prediction")
plt.xlabel("Salary")
plt.ylabel("Years of experience")
plt.legend()
plt.show()
```