Linear_Regressian

November 17, 2024

1 Salary Prediction

in the current era, the issue of employee salaries is very important. Because salary are used to meet the needs of life and increase employee motivation to work. Therefore, a payroll staff assigns a data analyst to estimate the cost must be incurred by the company to provide employee salaries that are appropriate and in accordance with employee achievement.

Regression analysis is the most widely used method of prediction, for regressian analysis, the first step is import library

[1]:
```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import plotly.express as px
import plotly.graph_objs as go
from plotly.offline import iplot
```

[2]:
```
#importing dataset
dataset = pd.read_csv('Salary_Data.csv')
```

[3]:
```
#to ensure that the imported is correct
dataset.head()
```

[3]:

	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

[4]:
```
#To see first 5 rows of the dataset
dataset.head().style.background_gradient(cmap = 'pink_r')
```

[4]: <pandas.io.formats.style.Styler at 0x1e8ca14beb0>

[5]:
```
#To see information of dataset
dataset.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
```

[6]:
```
dataset.isnull().sum().sum()
```

[6]: 0

it can be seen that are no missing values

[7]:
```
#Statistical Analysis
dataset.describe().style.background_gradient(cmap='pink_r')
```

[7]: <pandas.io.formats.style.Styler at 0x1e8ca14b880>

from the information above it can be seen that there are no outlierss

The relationship between variables can be seen by using sns.pairplot(). For multiple linear regression. it will be very useful, as it shows each feature with the responses using a heat map

1.1 Heat Map

[8]:
```
fig = px.imshow(dataset.corr())
fig.show()
```

1.1.1 Visualization

1.1.2 Scatter Plot

[9]:
```
scatter = [go.Scatter(x = dataset['YearsExperience'],y = dataset['Salary'],␣
 ↪mode='markers')]
fig = go.Figure(scatter)

iplot(fig)
```

[10]:
```
hist = [go.Histogram(x = dataset['YearsExperience'],\
                     marker=dict(color ='#AFE400',line =␣
 ↪dict(color='black',width=2)))]
```

[11]:
```
fig = go.Figure(data = hist)
iplot(fig)
```

[12]:
```
hist = [go.Histogram(x = dataset['Salary'],\
                     marker=dict(color ='#0FE400',line =␣
 ↪dict(color='black',width=2)))]

fig = go.Figure(data = hist)

iplot(fig)
```

[13]:
```
#Assigning dependent variable to y and independent variabel to X
```

[14]:
```
x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

[15]:
```
print(x)
```

```
[[ 1.1]
 [ 1.3]
 [ 1.5]
 [ 2. ]
 [ 2.2]
 [ 2.9]
 [ 3. ]
 [ 3.2]
 [ 3.2]
 [ 3.7]
 [ 3.9]
 [ 4. ]
 [ 4. ]
 [ 4.1]
 [ 4.5]
 [ 4.9]
 [ 5.1]
 [ 5.3]
 [ 5.9]
 [ 6. ]
 [ 6.8]
 [ 7.1]
 [ 7.9]
 [ 8.2]
 [ 8.7]
 [ 9. ]
 [ 9.5]
 [ 9.6]
 [10.3]
 [10.5]]
```

[16]:
```
print(y)
```

```
[ 39343.  46205.  37731.  43525.  39891.  56642.  60150.  54445.  64445.
  57189.  63218.  55794.  56957.  57081.  61111.  67938.  66029.  83088.
  81363.  93940.  91738.  98273. 101302. 113812. 109431. 105582. 116969.
 112635. 122391. 121872.]
```

[17]:
```
# The dataset has to be split into a training set and a test set analysis. This␣
↪can be done by the function train_test_split function from the␣
↪model_selection module
#module of the Scikit-learn library
```

[18]:
```
#Spliting testdata into X_train, Xtrain, y_train,y_test

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.
↪33,random_state=42)
```

[19]:
```
#Now the data set will be divided into x_train,x_test,
#y_train,y_ytest based on the test_size we have provided as input.
# Here dataset on
```

[20]:
```
print(x_train)
```

```
[[ 2.2]
 [ 5.1]
 [ 2.9]
 [ 4.1]
 [ 4. ]
 [ 7.9]
 [ 1.3]
 [ 1.5]
 [ 9. ]
 [ 2. ]
 [ 7.1]
 [ 9.5]
 [ 5.9]
 [10.5]
 [ 6.8]
 [ 3.2]
 [ 3.9]
 [ 4.5]
 [ 6. ]
 [ 3. ]]
```

[21]:
```
print(x_test)
```

```
[[ 9.6]
 [ 4.9]
 [ 8.2]
 [ 5.3]
```

```
 [ 3.2]
 [ 3.7]
 [10.3]
 [ 8.7]
 [ 4. ]
 [ 1.1]]
```

[22]:
```
print(y_train)
```

```
[ 39891.  66029.  56642.  57081.  55794. 101302.  46205.  37731. 105582.
  43525.  98273. 116969.  81363. 121872.  91738.  54445.  63218.  61111.
  93940.  60150.]
```

using a linear regressian to predict

[23]:
```
from sklearn.linear_model import LinearRegression
```

[24]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train, y_train)
```

[24]:
```
LinearRegression()
```

[25]:
```
y_pred = lr.predict(x_test)
```

[26]:
```
x_range = np.linspace(x.min(), x.max(), 100)
y_range = lr.predict(x_range.reshape(-1, 1))

fig = go.Figure([
        go.Scatter(x=x_train.squeeze(), y=y_train,
                   name='train', mode='markers'),
        go.Scatter(x=x_test.squeeze(), y=y_test,
                   name='test', mode='markers'),
        go.Scatter(x=x_range, y=y_range,
                   name='prediction')
    ])

fig.show()
```

[27]:
```
#Assigning Cofficient (slope) to b

b = lr.coef_
```

[28]:
```
print("Coefficient: ", b)
```

```
Coefficient:  [9426.03876907]
```

[29]:
```
a = lr.intercept_
```

[30]:
```
print("Intercept: ", a)
```

```
Intercept:  25324.335379244316
```

[31]:
```
#For this model, the linear regression equation will be predicting
#For years of experience 11, predicted salary can be calculated as:
```

[32]:
```
print(lr.predict([[11]]))
```

```
[129010.76183907]
```

[33]:
```
#Evaluation
#Mean Squared Error (MSE)
from sklearn import metrics
```

[34]:
```
print('Mean Squared Error (MSE): ', metrics.mean_squared_error(y_test, y_pred))
```

```
Mean Squared Error (MSE):  35301898.88713492
```

[35]:
```
import statsmodels.api as sm
```

[36]:
```
x_stat = sm.add_constant(x_train)
regsummary = sm.OLS(y_train, x_stat).fit()
regsummary.summary()
```

[36]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.955
Model:                            OLS   Adj. R-squared:                  0.952
Method:                 Least Squares   F-statistic:                     381.3
Date:                Sun, 17 Nov 2024   Prob (F-statistic):           1.45e-13
Time:                        22:58:17   Log-Likelihood:                -200.48
No. Observations:                  20   AIC:                             405.0
Df Residuals:                      18   BIC:                             406.9
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       2.532e+04   2743.538      9.231      0.000    1.96e+04    3.11e+04
x1          9426.0388    482.706     19.527      0.000    8411.911    1.04e+04
==============================================================================
Omnibus:                        0.822   Durbin-Watson:                   1.772
Prob(Omnibus):                  0.663   Jarque-Bera (JB):                0.819
Skew:                           0.380   Prob(JB):                        0.664
Kurtosis:                       2.363   Cond. No.                         12.4
==============================================================================

Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[37]:
```python
print("Adjusted R-Square : ",regsummary.rsquared_adj)
print("R-Square : ",regsummary.rsquared)
```

```
Adjusted R-Square :  0.9524194554302405
R-Square :  0.9549236946181227
```

[38]:
```python
from sklearn.metrics import r2_score
```

[39]:
```python
r2_score(y_train, lr.predict(x_train))
```

[39]:
```
0.9549236946181227
```

From data analysis that been carried out it can be concluded such as: 1. Variabel Years Experienced and Salary have a very strong correlation

2. The salary given by a payrollstaff to an employee who works for 11 years is between 129009.8069 USD to 129011.761762 USD