rxupn6g9w

March 4, 2024

1 Name: Sawant Rutuja Sanjay

2 Roll No.: TCOD79

3 Batch: T12

[47]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import linear_model
```

[48]:
```
data = {
    'age': [25, 30, 35, 40, 45, 50],
    'salary': [50000, 60000, 75000, 90000, 105000, 120000]
}

df = pd.DataFrame(data)
```

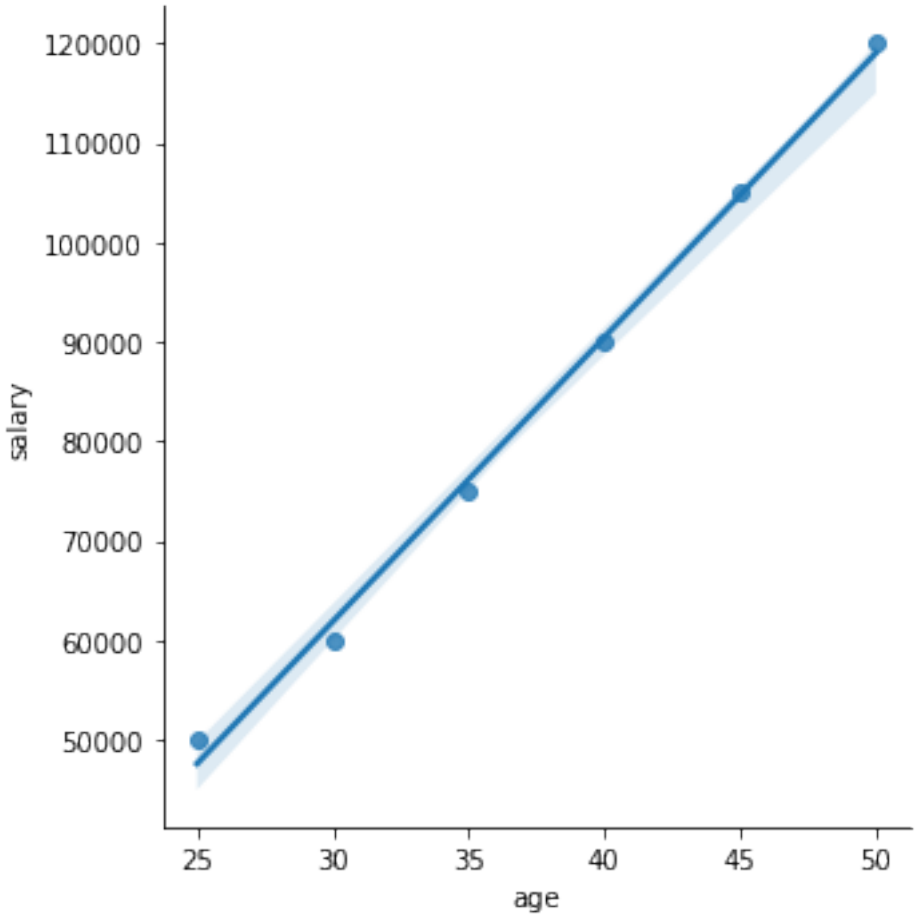
[49]:
```
df
```

[49]:

	age	salary
0	25	50000
1	30	60000
2	35	75000
3	40	90000
4	45	105000
5	50	120000

[50]:
```
sns.lmplot(x = 'age', y='salary', data = df)
```

[50]:
```
<seaborn.axisgrid.FacetGrid at 0x293cc838c40>
```



[51]:
```
reg = linear_model.LinearRegression()
```

[52]:
```
reg.fit(df[['age']],df['salary'])
```

[52]:
```
LinearRegression()
```

[53]:
```
reg.predict([[55]])
```

```
c:\users\rutuj\appdata\local\programs\python\python38\lib\site-
packages\sklearn\base.py:465: UserWarning: X does not have valid feature names,
but LinearRegression was fitted with feature names
  warnings.warn(
```

[53]:
```
array([133333.33333333])
```

[54]:
```
reg.predict([[58]])
```

```
c:\users\rutuj\appdata\local\programs\python\python38\lib\site-
packages\sklearn\base.py:465: UserWarning: X does not have valid feature names,
```

```
but LinearRegression was fitted with feature names
  warnings.warn(
```

[54]:
```
array([141904.76190476])
```

[55]:
```
reg.coef_
```

[55]:
```
array([2857.14285714])
```

[56]:
```
reg.intercept_
```

[56]:
```
-23809.523809523802
```

[57]:
```
# for Boston dataset

df = pd.read_csv("housing.csv")
df
```

[57]:

	RM	LSTAT	PTRATIO	MEDV
0	6.575	4.98	15.3	504000.0
1	6.421	9.14	17.8	453600.0
2	7.185	4.03	17.8	728700.0
3	6.998	2.94	18.7	701400.0
4	7.147	5.33	18.7	760200.0
..	…	…	…	…
484	6.593	9.67	21.0	470400.0
485	6.120	9.08	21.0	432600.0
486	6.976	5.64	21.0	501900.0
487	6.794	6.48	21.0	462000.0
488	6.030	7.88	21.0	249900.0

[489 rows x 4 columns]

[58]:
```
df.dropna(inplace=True)
```

[59]:
```
df
```

[59]:

	RM	LSTAT	PTRATIO	MEDV
0	6.575	4.98	15.3	504000.0
1	6.421	9.14	17.8	453600.0
2	7.185	4.03	17.8	728700.0
3	6.998	2.94	18.7	701400.0
4	7.147	5.33	18.7	760200.0
..	…	…	…	…
484	6.593	9.67	21.0	470400.0
485	6.120	9.08	21.0	432600.0
486	6.976	5.64	21.0	501900.0

```
487  6.794   6.48     21.0  462000.0
488  6.030   7.88     21.0  249900.0

[489 rows x 4 columns]
```

[60]:
```
df.head()
```

[60]:
```
      RM  LSTAT  PTRATIO      MEDV
0  6.575   4.98     15.3  504000.0
1  6.421   9.14     17.8  453600.0
2  7.185   4.03     17.8  728700.0
3  6.998   2.94     18.7  701400.0
4  7.147   5.33     18.7  760200.0
```

[61]:
```
X = df[['RM', 'LSTAT', 'PTRATIO']]
Y = df[['MEDV']]
```

[114]:
```
from sklearn.model_selection import train_test_split    #used for arrays␣
 ↪splitting into train and tests for training & testing of machine learning␣
 ↪model
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size = 0.3,␣
 ↪random_state = 42)

# test_size = 0.3 means only 30% data will be used for testing and other 70%␣
 ↪data will be used for training
# random_state = If set to a fixed integer value, you'll get the same split␣
 ↪every time you run your code.
#                If you don't specify random_state, a new random seed will be␣
 ↪generated each time you run your code, leading to a different split.
```

[115]:
```
from sklearn.preprocessing import StandardScaler      # to standardize the␣
 ↪values of dataset into standard format
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

[116]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
```

[117]:
```
y_pred
```

[117]:
```
array([[347601.28410273],
       [533923.56352585],
       [432015.38401136],
       [226371.46731514],
```

```
[338442.28933378],
[410687.87905862],
[255085.6496009 ],
[765897.84407095],
[365512.18598877],
[625489.4525274 ],
[478155.58960358],
[373031.77709601],
[260061.90389556],
[258200.77439273],
[406729.07707269],
[555033.0881696 ],
[402677.55955888],
[375952.20064046],
[381319.020201  ],
[436460.66243619],
[480946.72465774],
[482776.20587298],
[384104.40286713],
[696779.75242921],
[490360.89171788],
[494761.72535742],
[525698.66552375],
[681265.98542682],
[734637.86857607],
[152050.14931311],
[546471.06083479],
[232217.72369473],
[566678.93736061],
[533951.60068523],
[302812.56960095],
[526743.52534659],
[678308.23837899],
[522897.02765304],
[720647.44288161],
[689495.22064833],
[436576.92771793],
[421644.30030991],
[319277.90920569],
[471993.11947203],
[398049.88663116],
[618045.35864736],
[352843.44563062],
[404146.59767472],
[424531.21952511],
[401680.29611158],
[274409.57316322],
```

```
[622034.54295942],
[606097.77041311],
[439560.29572181],
[529370.20803514],
[530348.83542711],
[377106.0941929 ],
[560394.78526469],
[454202.92006332],
[270161.10928495],
[500611.52436056],
[400483.06652747],
[549706.47299106],
[493756.35698279],
[610736.98091818],
[826975.83649629],
[561753.41757762],
[569839.51664   ],
[-23922.0890207 ],
[347720.60999755],
[276335.97813649],
[425489.34038702],
[438375.61276982],
[476972.96462289],
[356735.63289345],
[479145.12239571],
[486441.48759758],
[439620.62108872],
[546850.08689685],
[437093.81730695],
[540003.77787012],
[610173.22069401],
[382025.66381659],
[276223.13874097],
[501213.05635307],
[606782.58042444],
[455598.25109436],
[526090.10734427],
[396951.66644033],
[431375.83134448],
[480451.19921116],
[418912.94300964],
[387238.88171194],
[508096.66219818],
[410983.15377836],
[604123.14843743],
[484932.93550692],
[277203.31126978],
```

```
[437481.22612258],
[283142.10883807],
[445651.19714566],
[425466.78004405],
[559422.7913246 ],
[312878.98514786],
[402335.2225684 ],
[424219.36130668],
[441463.37342019],
[518580.41258243],
[567020.7788324 ],
[510792.94146296],
[332884.65232359],
[627452.21214249],
[779517.08270557],
[396852.32036269],
[177166.00930181],
[419468.62885315],
[795560.36761054],
[362427.24735889],
[346816.34566983],
[550632.45292944],
[428861.72824262],
[404695.08957327],
[430635.55701578],
[ 86308.90138749],
[346278.99453049],
[795369.8074184 ],
[455283.50172265],
[720979.16806097],
[683553.51037908],
[350877.63602793],
[376910.65881488],
[413103.7033676 ],
[513735.62083916],
[550373.92966967],
[723733.2085772 ],
[514013.84160007],
[589344.35679738],
[367977.08556007],
[297417.85404874],
[607065.03356857],
[455018.65691416],
[526821.1320113 ],
[584893.00859321],
[498102.073264  ],
[462940.98239431],
```

```
       [464240.1823053 ],
       [363171.87820006]])
```

[118]:
```
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Root Mead squared Error is:")
print(rmse)
```

```
Root Mead squared Error is:
89987.31170641544
```

[119]:
```
print("Training accuracy is:")
lr.score(X_train, y_train)
```

```
Training accuracy is:
```

[119]:
```
0.7220172075811035
```