ML_02

January 21, 2022

1 Machine Learning (Lecture-2)

1.1 Multiple Linear regression practice

1.1.1 multiple-variables problem

- One dependent and two or more independent variable
- y= 1x1+ 2x2+ + nxn+ 0
  - y: response variable
  - n: number of features
  - xn: n-th feature
  - n: regression coefficient (weight) of the n-th feature
  - 0: y-intercept

1.2 Two types of variables

1.2.1 Independent and dependent

- independent (featuers, input data,permutation feature)
- dependent (prediction, output, response variable)

1.3 Step-1

1.3.1 Import Libraries

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

2 Multi_linear regression (Case-1)

[ ]:
```
# load dataset
df = pd.read_csv("ml_data_salary.csv")
df.head()
```

[ ]:

	age	distance	YearsExperience	Salary
0	31.1	77.75	1.1	39343
1	31.3	78.25	1.3	46205

```
2  31.5     78.75              1.5   37731
3  32.0     80.00              2.0   43525
4  32.2     80.50              2.2   39891
```

[ ]:
```
features = ['age', 'distance', 'YearsExperience']
target = 'Salary'
X = df[features].values.reshape(-1, len(features))
y = df[target].values
ols = linear_model.LinearRegression()
model = ols.fit(X, y)
```

[ ]:
```
model.coef_
```

[ ]:
```
array([-2.79782201e+15,  1.10953700e+15,  2.39795093e+13])
```

[ ]:
```
model.intercept_
```

[ ]:
```
719385278130755.0
```

[ ]:
```
model.score(X, y)
```

[ ]:
```
0.9569431439493807
```

[ ]:
```
x_pred = np.array([40,78.8, 2])
x_pred = x_pred.reshape(-1, len(features))
```

[ ]:
```
x_pred = np.array([[33,77,5], [45, 81.1, 2.5]])
x_pred = x_pred.reshape(-1, len(features))
```

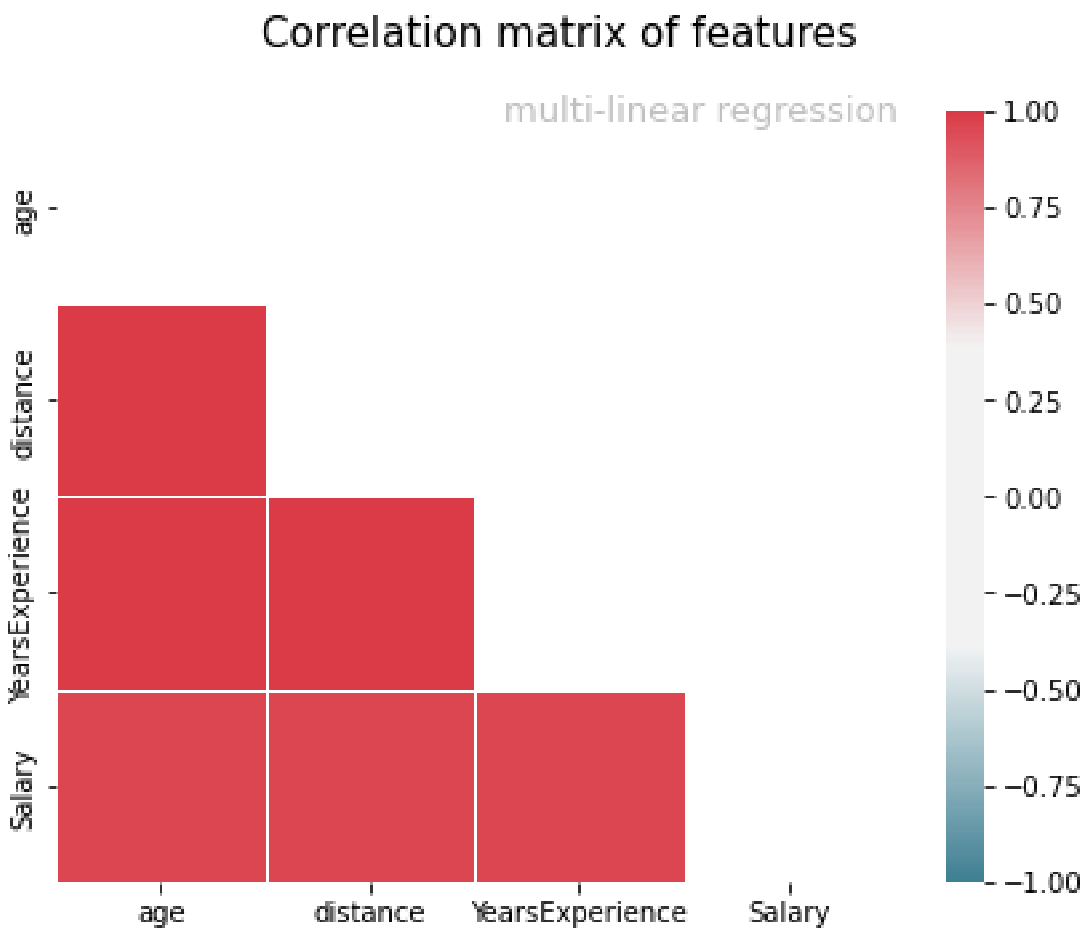
[ ]:
```
model.predict(x_pred)
```

[ ]:
```
array([-6.05449447e+15, -3.51392056e+16])
```

[ ]:
```
import seaborn as sns
import pandas as pd

corr = df.corr(method='spearman')
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
fig, ax = plt.pyplot.subplots(figsize=(6, 5))
cmap = sns.diverging_palette(220, 10, as_cmap=True, sep=100)
sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0, linewidths=.
 ↪5)

fig.suptitle('Correlation matrix of features', fontsize=15)
ax.text(0.77, 1, 'multi-linear regression', fontsize=13, ha='center',␣
 ↪va='center',
         transform=ax.transAxes, color='grey', alpha=0.5)
```

```
fig.tight_layout()
```

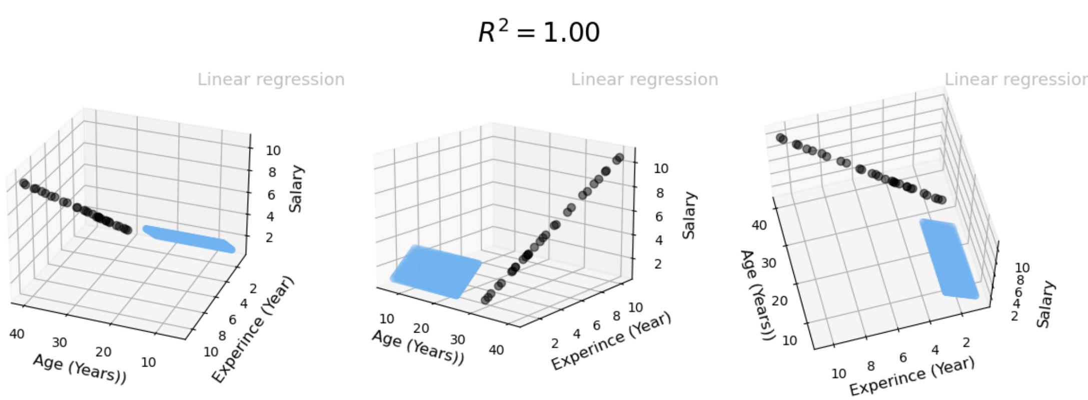
```
C:\Users\masha\AppData\Local\Temp/ipykernel_24908/2483135045.py:5:
DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To
silence this warning, use `bool` by itself. Doing this will not modify any
behavior and is safe. If you specifically wanted the numpy scalar type, use
`np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  mask = np.zeros_like(corr, dtype=np.bool)
```



[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from mpl_toolkits.mplot3d import Axes3D
```

```python
######################################## Data preparation␣
↪#########################################
features = ['age', 'YearsExperience']
target = 'Salary'
X = df[features].values.reshape(-1, len(features))
y = df[target].values


####################### Prepare model data point for visualization␣
↪###############################

x = X[:, 0]
y = X[:, 1]
z = y

x_pred = np.linspace(6, 24, 30)      # range of porosity values
y_pred = np.linspace(0.93, 2.9, 30)  # range of VR values
xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)
model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T

################################################ Train␣
↪#############################################

ols = linear_model.LinearRegression()
model = ols.fit(X, y)
predicted = model.predict(model_viz)

############################################## Evaluate␣
↪############################################

r2 = model.score(X, y)

############################################## Plot␣
↪################################################

plt.style.use('default')

fig = plt.figure(figsize=(12, 4))

ax1 = fig.add_subplot(131, projection='3d')
ax2 = fig.add_subplot(132, projection='3d')
ax3 = fig.add_subplot(133, projection='3d')

axes = [ax1, ax2, ax3]

for ax in axes:
```

```
    ax.plot(x, y, z, color='k', zorder=15, linestyle='none', marker='o',␣
 ↪alpha=0.5)
    ax.scatter(xx_pred.flatten(), yy_pred.flatten(), predicted,␣
 ↪facecolor=(0,0,0,0), s=20, edgecolor='#70b3f0')
    ax.set_xlabel('Age (Years))', fontsize=12)
    ax.set_ylabel('Experince (Year)', fontsize=12)
    ax.set_zlabel('Salary', fontsize=12)
    ax.locator_params(nbins=4, axis='x')
    ax.locator_params(nbins=5, axis='x')

ax1.text2D(1, 1, 'Linear regression', fontsize=13, ha='center', va='center',
           transform=ax1.transAxes, color='grey', alpha=0.5)
ax2.text2D(1, 1, 'Linear regression', fontsize=13, ha='center', va='center',
           transform=ax2.transAxes, color='grey', alpha=0.5)
ax3.text2D(1, 1, 'Linear regression', fontsize=13, ha='center', va='center',
           transform=ax3.transAxes, color='grey', alpha=0.5)

ax1.view_init(elev=27, azim=112)
ax2.view_init(elev=16, azim=-51)
ax3.view_init(elev=60, azim=165)

fig.suptitle('$R^2 = %.2f$' % r2, fontsize=20)

fig.tight_layout()
```



3 Multi_linear regression (Case-2)

3.0.1 without splitting in training and testing

```
[ ]: # load dataset
df1 = pd.read_csv("ml_data_salary.csv")
df1.head()
```

```
[ ]:     age  distance  YearsExperience  Salary
     0  31.1     77.75              1.1   39343
     1  31.3     78.25              1.3   46205
     2  31.5     78.75              1.5   37731
     3  32.0     80.00              2.0   43525
     4  32.2     80.50              2.2   39891
```

```
[ ]: X=df1[['age','distance','YearsExperience']]
     y=df1['Salary']
```

```
[ ]: #create and fit your model
     model_linear=LinearRegression().fit(X,y)
     model_linear
```

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

```
[ ]: model_linear.coef_
```

```
[ ]: array([-2.79782201e+15,  1.10953700e+15,  2.39795093e+13])
```

```
[ ]: model_linear.intercept_
```

```
[ ]: 719385278130755.0
```

```
[ ]: model_linear.score(X, y)
```

```
[ ]: 0.9569431439493807
```

```
[ ]: model_linear.predict([[31,77,5]])
```

```
C:\Anaconda\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have
valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

```
[ ]: array([-4.58850462e+14])
```

```
[ ]: model_linear.score(X,y)
```

```
[ ]: 0.9569431439493807
```

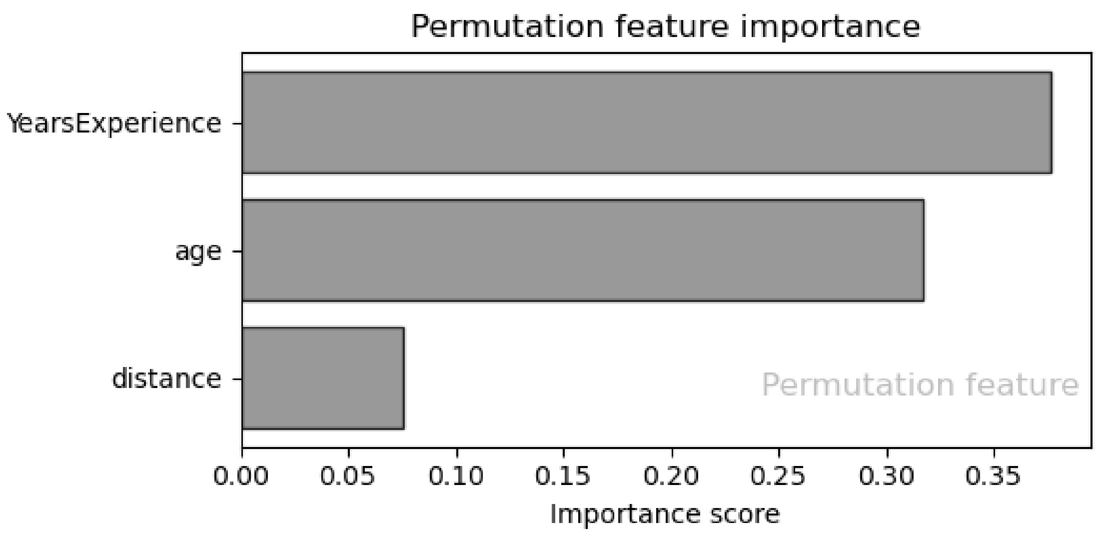
4 Multi_linear regression (Case-2)

4.0.1 Splitting in training and testing

```
[ ]: pip install rfpimp
```

```
[ ]: import rfpimp
     import pandas as pd
```

```
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split

######################################## Data preparation␣
↪#########################################

df3 = pd.read_csv("ml_data_salary.csv")
features = ['age', 'YearsExperience','distance','Salary']


######################################## Train/test split␣
↪#########################################

df3_train, df3_test = train_test_split(df3, test_size=0.20)
df3_train = df3_train[features]
df3_test = df3_test[features]

X_train, y_train = df3_train.drop('Salary',axis=1), df3_train['Salary']
X_test, y_test = df3_test.drop('Salary',axis=1), df3_test['Salary']

################################################ Train␣
↪#############################################

rf = RandomForestRegressor(n_estimators=100, n_jobs=-1)
rf.fit(X_train, y_train)

############################### Permutation feature importance␣
↪#####################################

imp = rfpimp.importances(rf, X_test, y_test)

############################################## Plot␣
↪################################################

fig, ax = plt.subplots(figsize=(6, 3))

ax.barh(imp.index, imp['Importance'], height=0.8, facecolor='grey', alpha=0.8,␣
↪edgecolor='k')
ax.set_xlabel('Importance score')
ax.set_title('Permutation feature importance')
ax.text(0.8, 0.15, 'Permutation feature', fontsize=12, ha='center', va='center',
        transform=ax.transAxes, color='grey', alpha=0.5)
plt.gca().invert_yaxis()

fig.tight_layout()
```

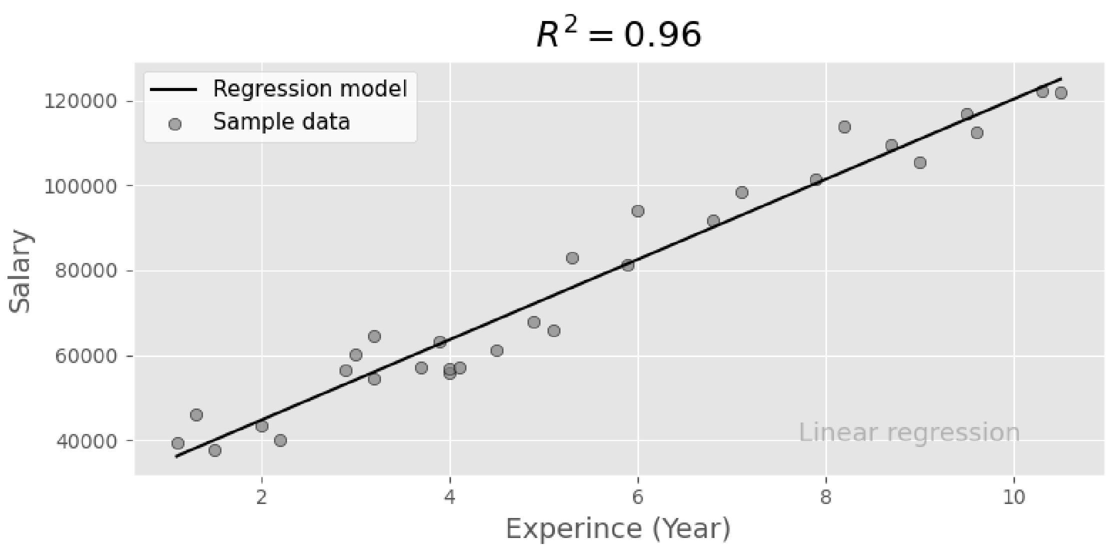


5 Permutation feature

5.0.1 Based on the permutation feature importances shown in figure, experince is the most important feature, and age is the second most important feature.

```
[ ]: import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model

######################################## Data preparation␣
↪#########################################
df3 = pd.read_csv("ml_data_salary.csv")

X = df3['YearsExperience'].values.reshape(-1,1)
y = df3['Salary'].values

################################################ Train␣
↪#############################################

ols = linear_model.LinearRegression()
model = ols.fit(X, y)
response = model.predict(X)

############################################## Evaluate␣
↪############################################

r2 = model.score(X, y)
```

```
################################################## Plot␣
↪##################################################

plt.style.use('default')
plt.style.use('ggplot')

fig, ax = plt.subplots(figsize=(8, 4))

ax.plot(X, response, color='k', label='Regression model')
ax.scatter(X, y, edgecolor='k', facecolor='grey', alpha=0.7, label='Sample␣
↪data')
ax.set_ylabel('Salary', fontsize=14)
ax.set_xlabel('Experince (Year)', fontsize=14)
ax.text(0.8, 0.1, 'Linear regression', fontsize=13, ha='center', va='center',
         transform=ax.transAxes, color='grey', alpha=0.5)
ax.legend(facecolor='white', fontsize=11)
ax.set_title('$R^2= %.2f$' % r2, fontsize=18)

fig.tight_layout()
```



[ ]:
```
#predictions
x_pred = np.array([[15]])
```

[ ]:
```
model.predict(x_pred)
```

[ ]:
```
array([167541.63502049])
```

6 Multi_linear regression (Case-3)

[ ]:
```
import pandas as pd
df2=pd.read_csv("ml_data_salary.csv")
df2=pd.DataFrame(df2,columns=['age','YearsExperience'])
df2['Salary']=pd.Series(y)
df2
```

[ ]:

	age	YearsExperience	Salary
0	31.1	1.1	39343
1	31.3	1.3	46205
2	31.5	1.5	37731
3	32.0	2.0	43525
4	32.2	2.2	39891
5	32.9	2.9	56642
6	33.0	3.0	60150
7	33.2	3.2	54445
8	33.2	3.2	64445
9	33.7	3.7	57189
10	33.9	3.9	63218
11	34.0	4.0	55794
12	34.0	4.0	56957
13	34.1	4.1	57081
14	34.5	4.5	61111
15	34.9	4.9	67938
16	35.1	5.1	66029
17	35.3	5.3	83088
18	35.9	5.9	81363
19	36.0	6.0	93940
20	36.8	6.8	91738
21	37.1	7.1	98273
22	37.9	7.9	101302
23	38.2	8.2	113812
24	38.7	8.7	109431
25	39.0	9.0	105582
26	39.5	9.5	116969
27	39.6	9.6	112635
28	40.3	10.3	122391
29	40.5	10.5	121872

[ ]:
```
import statsmodels.formula.api as smf
model_2= smf.ols(formula='Salary~age+YearsExperience',data=df2)
results_formula=model_2.fit()
results_formula.params
```

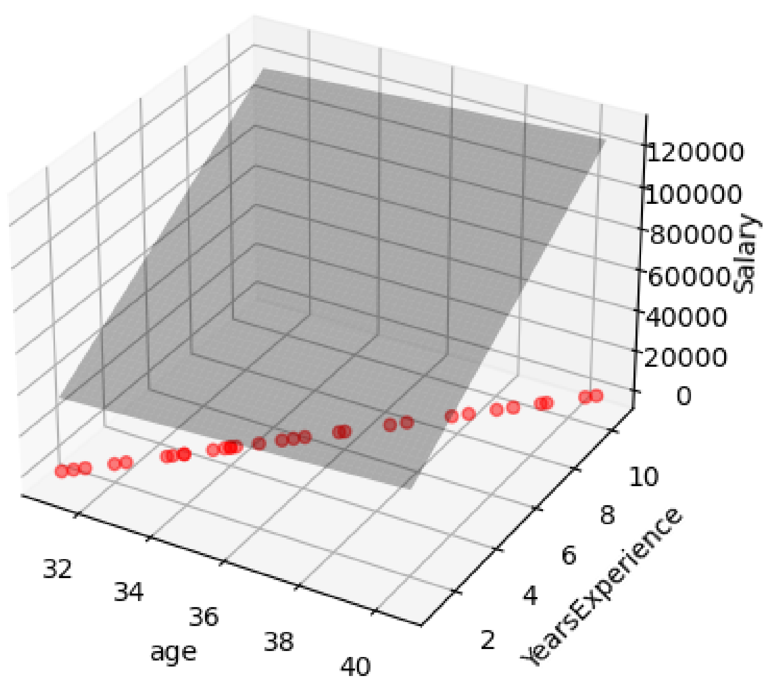
[ ]:
```
Intercept          -257.111385
age                 868.310386
```

```
YearsExperience    8581.651935
dtype: float64
```

[ ]:
```
x_surf,y_surf=np.meshgrid(np.linspace(df2.age.min(),df2.age.max(),100),np.
 ↪linspace(df2.YearsExperience.min(),df2.YearsExperience.max(),100))
onlyX=pd.DataFrame({'age':x_surf.ravel(),'YearsExperience':y_surf.ravel()})
fittedY= results_formula.predict(exog=onlyX)
```

[ ]:
```
fittedY=np.array(fittedY)
```

[ ]:
```
fig= plt.figure()
ax=fig.add_subplot(111,projection='3d')
ax.scatter(df2['age'],df2['YearsExperience'], c='red',marker='o',alpha=0.5)
ax.plot_surface(x_surf,y_surf,fittedY.reshape(x_surf.shape),color='None',␣
 ↪alpha=0.3)
ax.set_xlabel('age')
ax.set_ylabel('YearsExperience')
ax.set_zlabel('Salary')
plt.show()
```

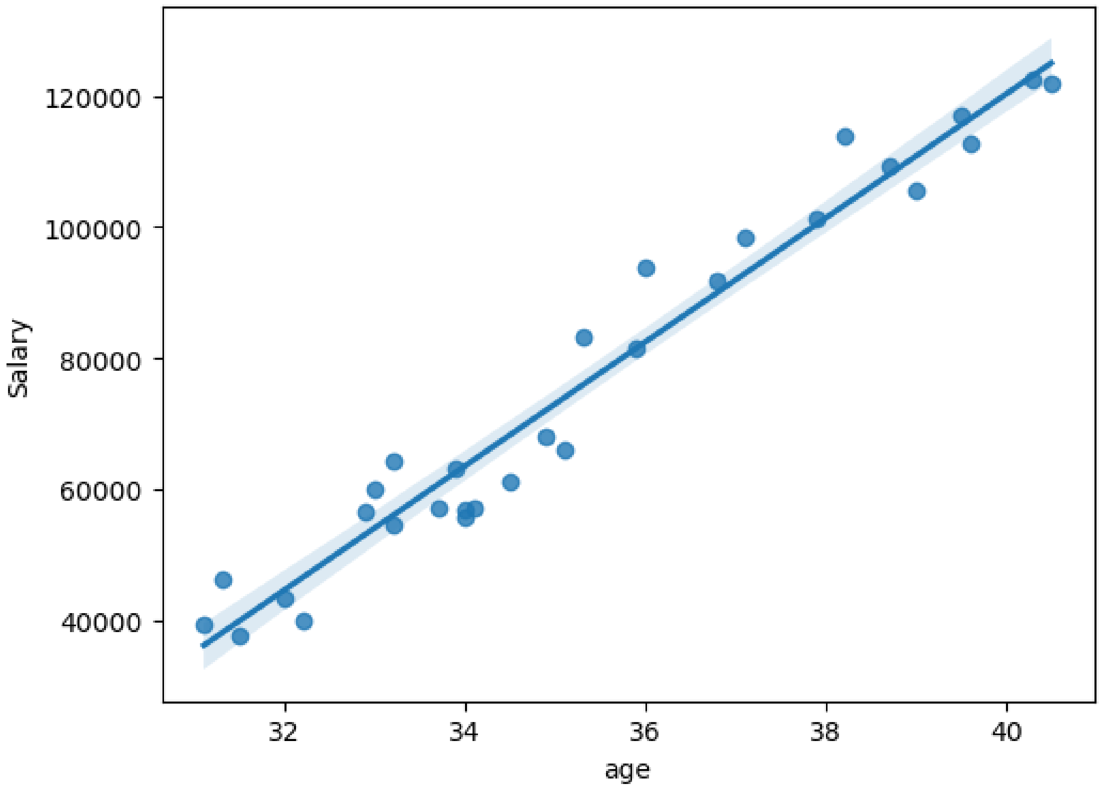


[ ]:
```
sns.regplot('age','Salary',data= df2)
```

```
C:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the
```

```
following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

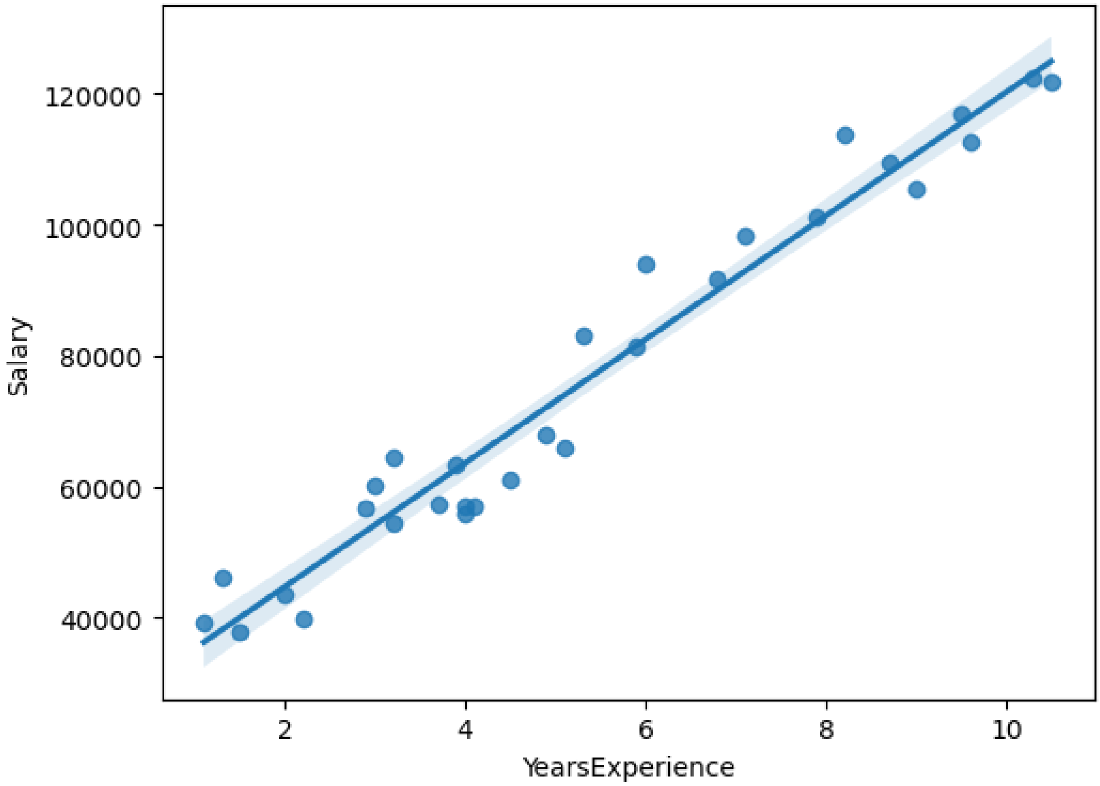
[ ]: <AxesSubplot:xlabel='age', ylabel='Salary'>



[ ]:
```
sns.regplot('YearsExperience','Salary',data= df2)
```

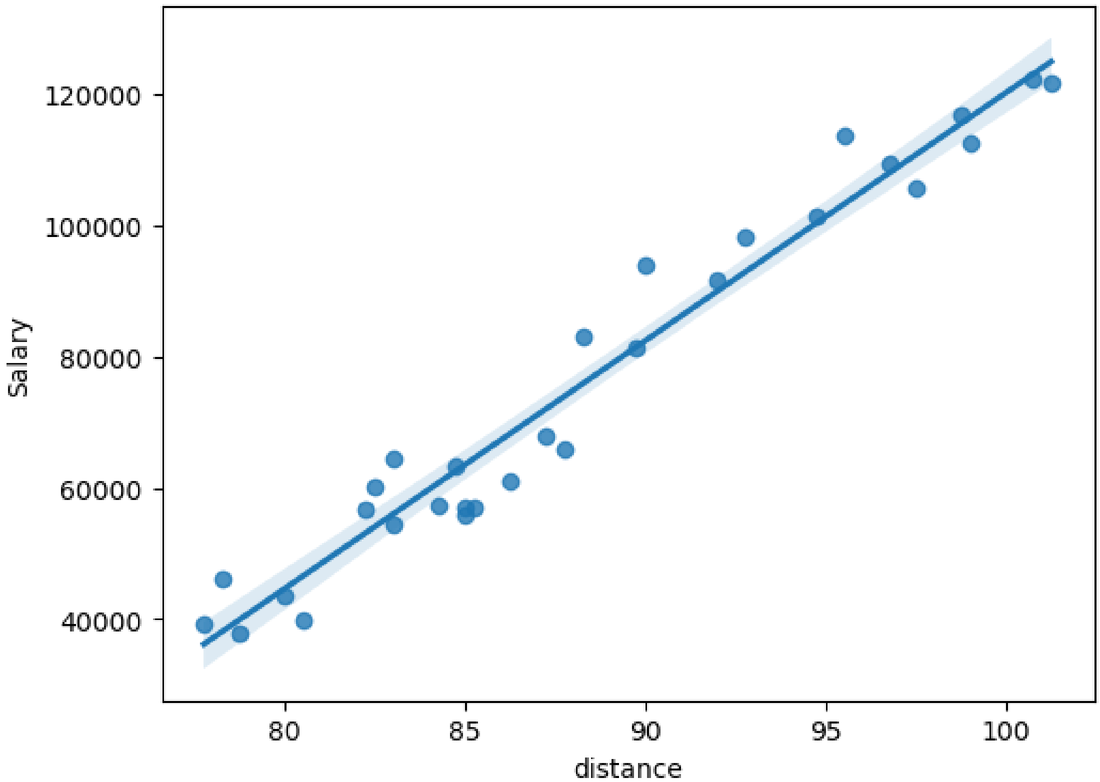
```
C:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

[ ]: <AxesSubplot:xlabel='YearsExperience', ylabel='Salary'>



[ ]:
```
sns.regplot('distance','Salary',data= df)
```

[ ]: <AxesSubplot:xlabel='distance', ylabel='Salary'>


120000
100000
80000
60000
40000
Salary
80
85
90
95
100
distance