Machine learnig

Linear regression

- y = a + bx + b1 x1 + b2 x2 .....
- y => dependent / target (1) [1D]
- x => independent / features (n) [2D]

```
from sklearn.linear_model import LinearRegression
import numpy as np
from sklearn.metrics import r2_score,
mean_absolute_error,mean_squared_error

#independent
time=np.array([5,7,12,16,20]).reshape(-1,1)

#dependent
mass=np.array([40,120,180,210,240])

mymodel = LinearRegression()

#fit-used to trains  the model
mymodel.fit(time,mass)
```

```
LinearRegression()
```

```
x=int(input("Enter the time in minutes : "))
result = mymodel.predict([[x]])#predict ehich predict the
output ,passing int var (time 2D)
print("if the time is ",x," minutes the mass is ",result[0]," grams ")
```

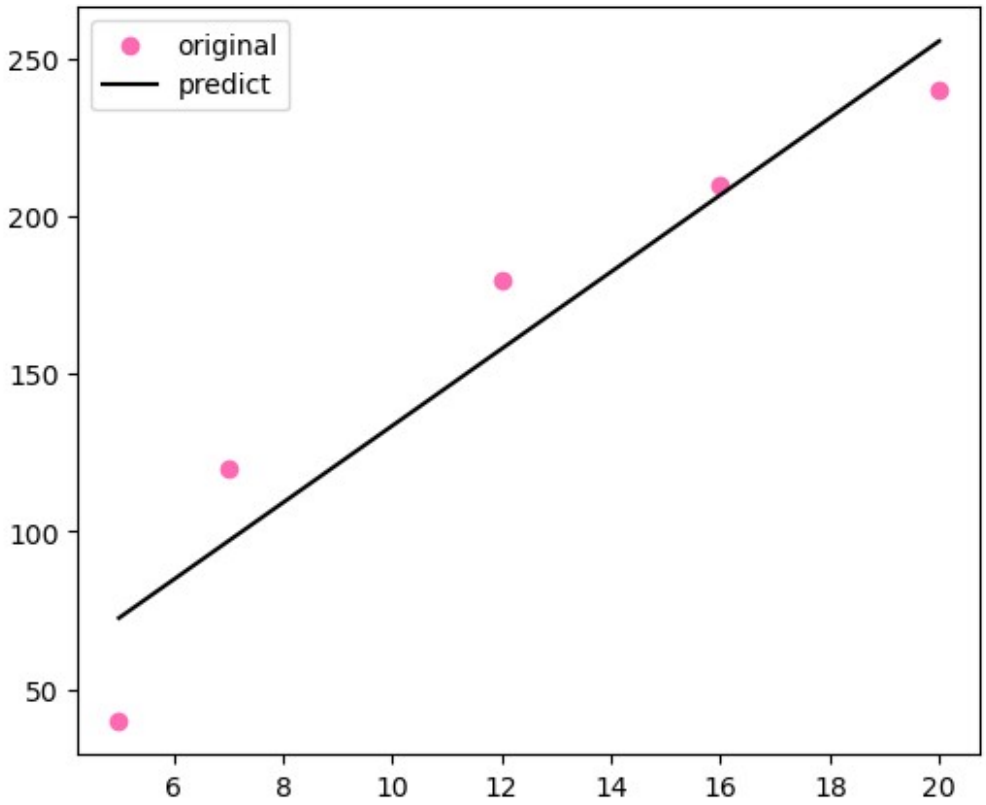
```
Enter the time in minutes :  4
```

```
if the time is  4  minutes the mass is  60.33766233766232  grams
```

```
mass_model = mymodel.predict(time)
print(mass_model)
```

```
[ 72.54545455  96.96103896 158.         206.83116883 255.66233766]
```

```
#plotting original values - scatter
import matplotlib.pyplot as plt
plt.figure(figsize=(6,5))
plt.scatter(time,mass,label='original',color='hotpink')
#plotting predict values in lines
plt.plot(time,mass_model,label='predict',color='k')
plt.legend()
plt.show()
```



Libraries -> load data -> split data -> create & train -> test -> Evaluate

split- train predict

Evaluation : R-square

- Lower , the better

```
r2s = r2_score(mass,mass_model)
print(r2s)#result is more than 0.5 then the result prediction are good
```

```
0.9078579648199901
```

MSE

- lower the better

```
mse = mean_squared_error(mass,mass_model)
print(mse)#biggest values in that
```

```
465.87012987012986
```

MAE

- lower the better

```
mae = mean_absolute_error(mass,mass_model)
print(mae)
```

```
19.283116883116882
```

linear regression on large data , only accepects the numerical values

case study : predicting the salary from age ,experiance,genedr ,education

1. import libraries
2. load data
3. clean data(nulls , duplicates)
4. preprocessing
5. split data
6. create and train model
7. test the model
8. Evaluation

1. importing the libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score , mean_absolute_error ,
mean_squared_error
from sklearn.model_selection import train_test_split
```

load the data

```
df=pd.read_csv(r"C:\Users\DELL\Downloads\my_python\Salary_EDA.csv")
df.head()
```

```
    Age  Gender Education Level          Job Title  Years of
Experience  \
0  32.0    Male      Bachelor's  Software Engineer
5.0
1  28.0  Female        Master's       Data Analyst
3.0
2  45.0    Male             PhD     Senior Manager
15.0
3  36.0  Female      Bachelor's    Sales Associate
7.0
4  36.0  Female      Bachelor's    Sales Associate
7.0

     Salary
0   90000.0
```

```
1   65000.0
2  150000.0
3   60000.0
4   60000.0
```

clean data

```
df.isnull().sum()
```

```
Age                    2
Gender                 4
Education Level        3
Job Title              5
Years of Experience    2
Salary                 3
dtype: int64
```

```
df.dropna(inplace = True )
df.isnull().sum()
```

```
Age                    0
Gender                 0
Education Level        0
Job Title              0
Years of Experience    0
Salary                 0
dtype: int64
```

data preprocessing

```
ge=LabelEncoder()
df['gendercode']=ge.fit_transform(df['Gender'])
#encoding edu_level
edu=LabelEncoder()
df['edu_code']=edu.fit_transform(df['Education Level'])
df.head()
```

```
    Age  Gender Education Level          Job Title  Years of
Experience  \
0  32.0    Male      Bachelor's  Software Engineer
5.0
1  28.0  Female        Master's       Data Analyst
3.0
2  45.0    Male             PhD     Senior Manager
15.0
3  36.0  Female      Bachelor's    Sales Associate
7.0
4  36.0  Female      Bachelor's    Sales Associate
7.0
```

```
     Salary  gendercode  edu_code
0   90000.0           1         0
1   65000.0           0         1
2  150000.0           1         2
3   60000.0           0         0
4   60000.0           0         0
```

split the data into independent and dependent

```
# x- independent , y- dependent
x = df[['Age','gendercode','edu_code','Years of Experience']] # [ 2D]
y = df['Salary'] # [1D]
```

split the data into train and test

```
x_train , x_test , y_train , y_test =
train_test_split(x,y,test_size=0.2, random_state = 42 )
#total 700 records
#x_train - 560(age ,edu,gen), train  paper
#x_test - 140(age,edu,gen), question paper
#y_train - 560(sal),train paper answer
#y_test - 140(sal) , answer paper
```

create a model and train

```
sal_model = LinearRegression()
sal_model.fit(x_train,y_train)

LinearRegression()
```

Test

```
a=float(input("enter the age"))
g_user= input("enter your gender(Male,Female) :")
ed_user= input(" enter your education level (Bachelor's,Master's,PhD):
")
exp= float(input("enter your experiance in your :"))

enter the age 45
enter your gender(Male,Female) : Male
 enter your education level (Bachelor's,Master's,PhD):  PhD
enter your experiance in your : 8

gen_enc=ge.transform([g_user])[0]
edu_enc=edu.transform([ed_user])[0]
print(gen_enc,edu_enc)

1 2
```

```
result=sal_model.predict([[a,gen_enc,edu_enc,exp]])
print("the predivted salary is : ",result[0])
```

```
the predivted salary is :  136557.9697061547
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\base.py:439:
UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(
```

Evaluation

- predict test values
- visulaize
- metrics

```
model_p= sal_model.predict(x_test)
```

```
len(y_test)
```

```
74
```

```
#plotting original values
import matplotlib.pyplot as plt
plt.figure(figsize=(6,5))
plt.plot(np.arange(1,75),y_test,label='original',color='hotpink')
#plotting predict values in lines
plt.plot(np.arange(1,75),model_p,label='predict',color='k')
plt.title("salary prediction")
plt.legend()
plt.show()
```


salary prediction
original
predict
250000
200000
150000
100000
50000
0
10
20
30
40
50
60
70