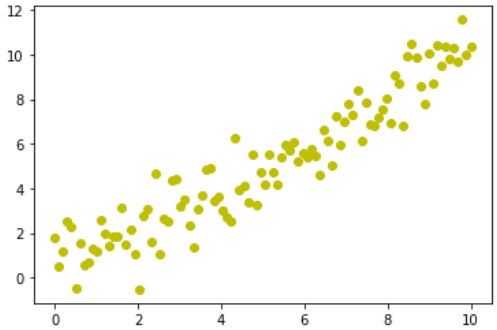
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Regression with SKLEARN

```
np.random.seed(0)
m = 100 # creating 100 sample
X = np.linspace(0,10,m).reshape(m,1)
y = X + np.random.randn(m,1)
```

```
plt.scatter(X,y,c='y')
```

```
<matplotlib.collections.PathCollection at 0x16c247d0a90>
```



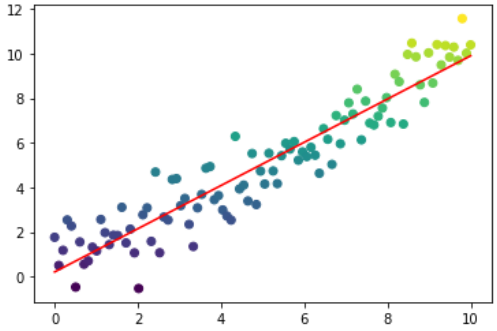
```
from sklearn.linear_model import LinearRegression
```

```
model = LinearRegression()
model.fit(X,y)
model.score(X,y)

predictions = model.predict(X)

plt.scatter(X,y,c=y)
plt.plot(X,predictions, c = 'r')
```

```
[<matplotlib.lines.Line2D at 0x16c2493c310>]
```



Classificaion problem with with SKLEARN

```
titanic = sns.load_dataset('titanic')
titanic.shape
titanic.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

```
titanic = titanic[['survived','pclass','sex','age']]
titanic.dropna(axis=0 , inplace=True)
titanic['sex'].replace(['male','female'] , [0,1] , inplace=True)
titanic.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	survived	pclass	sex	age
0	0	3	0	22.0
1	1	1	1	38.0
2	1	3	1	26.0
3	1	1	1	35.0
4	0	3	0	35.0

```
from sklearn.neighbors import KNeighborsClassifier
```

```
model = KNeighborsClassifier()
```

```
y = titanic['survived']
X = titanic.drop('survived' , axis=1)
```

```
X.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}
```

```
.dataframe thead th {
    text-align: right;
}
```

	pclass	sex	age
0	3	0	22.0
1	1	1	38.0
2	3	1	26.0
3	1	1	35.0
4	3	0	35.0

```
y.head()
```

```
0    0
1    1
2    1
3    1
4    0
Name: survived, dtype: int64
```

```
model.fit(X,y)
print(f'{model.score(X,y)*100} %')
```

```
84.17366946778712 %
```

The function which will say that if we will survive or not

```
def survived(model , pclass=3 , sex=0 , age=22):
    x = np.array([pclass,sex,age]).reshape(1,3)
    if model.predict(x) == [0]:
        print('You will not servive')
    else:
        print('You will survive')
```

```
survived(model)
```

```
You will not servive
```

The probability of surviving (vice versa)

```
def survived(model , pclass=3 , sex=0 , age=22):
    x = np.array([pclass,sex,age]).reshape(1,3)
    if model.predict(x) == [0]:
        print('You will not servive')
        print(f'you will not survive with {model.predict_proba(x)[0,0]*100}% and survive with {model.predict_proba(x)
[0,1]*100}%')
    else:
        print('You will survive')
        print(f'you will not survive with {model.predict_proba(x)[0,0]*100}% and survive with {model.predict_proba(x)
[0,1]*100}%')
```

```
survived(model,3,1,25)
```

```
You will not servive
you will not survive with 80.0% and survive with 20.0%
```