Double-click (or enter) to edit

```
*Roll* No : 1233

Name : Kirti Vichare
```

Roll No : 1233

Name : Kirti Vichare

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

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

Double-click (or enter) to edit

```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

Double-click (or enter) to edit

```
dataset = pd.read_csv('/content/drive/MyDrive/Social_Network_Ads.csv')
```

Double-click (or enter) to edit

```
dataset
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1

```
#display no. of rows and columns
#columnns
dataset.columns
```

```
Index(['User ID', 'Gender', 'Age', 'EstimatedSalary', 'Purchased'], dtype='object')
```

400 rows × 5 columns

```
#rows
dataset.shape
```

```
(400, 5)
```

```
#dataset info
dataset.info
```

```
<bound method DataFrame.info of       User ID  Gender  Age  EstimatedSalary  Purchase
0    15624510    Male   19            19000         0
3    15603246  Female   27            57000         0
4    15804002    Male   19            76000         0
..        ...     ...  ...              ...       ...
395  15691863  Female   46            41000         1
396  15706071    Male   51            23000         1
397  15654296  Female   50            20000         1
398  15755018    Male   36            33000         0
399  15594041  Female   49            36000         1

[400 rows x 5 columns]>
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
#describe
dataset.describe()
```

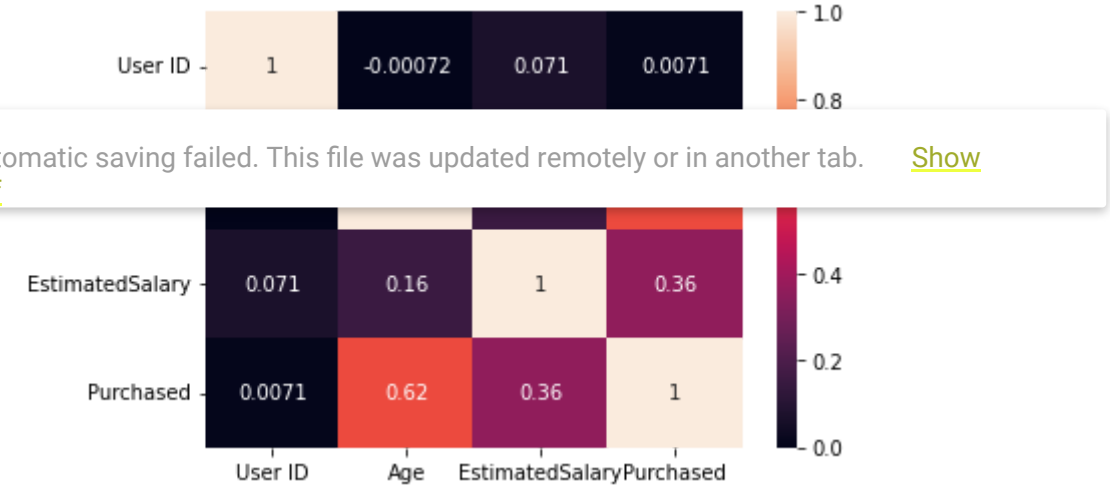
	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000

```
dataset.corr()
```

	User ID	Age	EstimatedSalary	Purchased
User ID	1.000000	-0.000721	0.071097	0.007120
Age	-0.000721	1.000000	0.155238	0.622454
EstimatedSalary	0.071097	0.155238	1.000000	0.362083
Purchased	0.007120	0.622454	0.362083	1.000000

```
import seaborn as sn
sn.heatmap(dataset.corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3fe245d9d0>
```



Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
dataset.Purchased.unique()
```

```
array([0, 1])
```

```
x=dataset.iloc[:,[2,3]].values #age,estimated salary
y =dataset.iloc[:,4].values
```

Double-click (or enter) to edit

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=1)


from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train =sc.fit_transform(X_train)
X_test = sc.transform(X_test)


logreg = LogisticRegression(random_state=0)
logreg.fit(X_train,y_train)
```

```
LogisticRegression(random_state=0)
```

```
y_pred=logreg.predict(X_test)


from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[52  6]
 [11 31]]
```

Precison

```
precision = cm[0,0]/(cm[0,0]+cm[1,0])
print(precision)
```

```
0.8253968253968254
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
print(recall)
```

```
0.896551724137931
```

```
accuracy = (cm[0,0]+cm[1,1])/cm.sum()
print(accuracy)
```

```
0.83
```

```
y_train_pred = logreg.predict(X_train)


cm=confusion_matrix(y_test,y_pred)


precision = cm[0,0]/(cm[0,0]+cm[1,0])
print(precision)
```

```
0.8253968253968254
```

```
recall=cm[0,0]/(cm[0,0]+cm[0,1])
print(recall)
```

```
0.896551724137931
```

```
accuracy = (cm[0,0]+cm[1,1])/cm.sum()
print(accuracy)
```

```
0.83
```

```
from sklearn.metrics import accuracy_score
print("Testing Accuracy=" + str(accuracy_score(y_test,y_pred)))
print("Training Accuracy =" +str(accuracy_score( y_train,y_train_pred)))
```

```
Testing Accuracy=0.83
Training Accuracy =0.8433333333333334
```

LOG LOSS

```
from sklearn.metrics import log_loss
print("Log Loss Training =" + str(log_loss(y_test,y_pred)))
print("Log Loss Training =" +str(log_loss( y_train,y_train_pred)))
```

```
Log Loss Training =5.871639962980628
Log Loss Training =5.411112283082749
```

```
#Paramater tunning
param_grid = { 'C' : [0.5,0.8,1.0],
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(LogisticRegression(),param_grid, refit= True, verbose =3)
```

```
grid.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 30 candidates, totalling 150 fits
[CV 1/5] END C=0.5, multi_class=auto, solver=newton-cg;, score=0.817 total time=
[CV 2/5] END C=0.5, multi_class=auto, solver=newton-cg;, score=0.850 total time=
[CV 3/5] END C=0.5, multi_class=auto, solver=newton-cg;, score=0.817 total time=
[CV 4/5] END C=0.5, multi_class=auto, solver=newton-cg;, score=0.850 total time=
[CV 5/5] END C=0.5, multi_class=auto, solver=newton-cg;, score=0.850 total time=
[CV 1/5] END C=0.5, multi_class=auto, solver=lbfgs;, score=0.817 total time=   0.0
[CV 2/5] END C=0.5, multi_class=auto, solver=lbfgs;, score=0.850 total time=   0.0
[CV 3/5] END C=0.5, multi_class=auto, solver=lbfgs;, score=0.817 total time=   0.0
[CV 4/5] END C=0.5, multi_class=auto, solver=lbfgs;, score=0.850 total time=   0.0
[CV 5/5] END C=0.5, multi_class=auto, solver=lbfgs;, score=0.850 total time=   0.0
[CV 1/5] END C=0.5, multi_class=auto, solver=liblinear;, score=0.817 total time=
[CV 2/5] END C=0.5, multi_class=auto, solver=liblinear;, score=0.850 total time=
[CV 3/5] END C=0.5, multi_class=auto, solver=liblinear;, score=0.800 total time=
[CV 4/5] END C=0.5, multi_class=auto, solver=liblinear;, score=0.867 total time=
```

```
[CV 5/5] END C=0.5, multi_class=auto, solver=liblinear;, score=0.867 total time=
[CV 1/5] END C=0.5, multi_class=auto, solver=sag;, score=0.817 total time=   0.0s
[CV 2/5] END C=0.5, multi_class=auto, solver=sag;, score=0.850 total time=   0.0s
[CV 3/5] END C=0.5, multi_class=auto, solver=sag;, score=0.817 total time=   0.0s
[CV 4/5] END C=0.5, multi_class=auto, solver=sag;, score=0.850 total time=   0.0s
[CV 5/5] END C=0.5, multi_class=auto, solver=sag;, score=0.850 total time=   0.0s
[CV 1/5] END C=0.5, multi_class=auto, solver=saga;, score=0.817 total time=   0.0s
[CV 2/5] END C=0.5, multi_class=auto, solver=saga;, score=0.850 total time=   0.0s
[CV 3/5] END C=0.5, multi_class=auto, solver=saga;, score=0.817 total time=   0.0s
[CV 4/5] END C=0.5, multi_class=auto, solver=saga;, score=0.850 total time=   0.0s
[CV 5/5] END C=0.5, multi_class=auto, solver=saga;, score=0.850 total time=   0.0s
[CV 1/5] END C=0.5, multi_class=ovr, solver=newton-cg;, score=0.817 total time=   (
[CV 2/5] END C=0.5, multi_class=ovr, solver=newton-cg;, score=0.850 total time=   (
[CV 3/5] END C=0.5, multi_class=ovr, solver=newton-cg;, score=0.817 total time=   (
[CV 4/5] END C=0.5, multi_class=ovr, solver=newton-cg;, score=0.850 total time=   (
[CV 5/5] END C=0.5, multi_class=ovr, solver=newton-cg;, score=0.850 total time=   (
[CV 1/5] END C=0.5, multi_class=ovr, solver=lbfgs;, score=0.817 total time=   0.0s
[CV 2/5] END C=0.5, multi_class=ovr, solver=lbfgs;, score=0.850 total time=   0.0s
[CV 3/5] END C=0.5, multi_class=ovr, solver=lbfgs;, score=0.817 total time=   0.0s
[CV 4/5] END C=0.5, multi_class=ovr, solver=lbfgs;, score=0.850 total time=   0.0s
[CV 5/5] END C=0.5, multi_class=ovr, solver=lbfgs;, score=0.850 total time=   0.0s
[CV 1/5] END C=0.5, multi_class=ovr, solver=liblinear;, score=0.817 total time=   (
[CV 2/5] END C=0.5, multi_class=ovr, solver=liblinear;, score=0.850 total time=   (
[CV 3/5] END C=0.5, multi_class=ovr, solver=liblinear;, score=0.800 total time=   (
[CV 4/5] END C=0.5, multi_class=ovr, solver=liblinear;, score=0.867 total time=   (
[CV 5/5] END C=0.5, multi_class=ovr, solver=liblinear;, score=0.867 total time=   (
[CV 1/5] END C=0.5, multi_class=ovr, solver=sag;, score=0.817 total time=   0.0s
[CV 2/5] END C=0.5, multi_class=ovr, solver=sag;, score=0.850 total time=   0.0s
[CV 3/5] END C=0.5, multi_class=ovr, solver=sag;, score=0.817 total time=   0.0s
[CV 4/5] END C=0.5, multi_class=ovr, solver=sag;, score=0.850 total time=   0.0s
[CV 5/5] END C=0.5, multi_class=ovr, solver=sag;, score=0.850 total time=   0.0s
[CV 1/5] END C=0.5, multi_class=ovr, solver=saga;, score=0.817 total time=   0.0s
[CV 2/5] END C=0.5, multi_class=ovr, solver=saga;, score=0.850 total time=   0.0s
[CV 3/5] END C=0.5, multi_class=ovr, solver=saga;, score=0.817 total time=   0.0s
[CV 4/5] END C=0.5, multi_class=ovr, solver=saga;, score=0.850 total time=   0.0s
[CV 5/5] END C=0.5, multi_class=ovr, solver=saga;, score=0.850 total time=   0.0s
[CV 1/5] END C=0.8, multi_class=auto, solver=newton-cg;, score=0.817 total time=
[CV 2/5] END C=0.8, multi_class=auto, solver=newton-cg;, score=0.850 total time=
[CV 3/5] END C=0.8, multi_class=auto, solver=newton-cg;, score=0.817 total time=
[CV 4/5] END C=0.8, multi_class=auto, solver=newton-cg;, score=0.850 total time=
[CV 5/5] END C=0.8, multi_class=auto, solver=newton-cg;, score=0.867 total time=
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
print(grid.best_params_)
```

```
{'C': 1.0, 'multi_class': 'auto', 'solver': 'liblinear'}
```

```
print(grid.best_estimator_)
```

```
LogisticRegression(solver='liblinear')
```

```
print('Logistic Regression',LogisticRegression(solver='liblinear'))
```

```
Logistic Regression LogisticRegression(solver='liblinear')
```

```
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
```

```
from sklearn.model_selection import cross_val_score
#evaluate Logistic regression

kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
cv_results = cross_val_score(LogisticRegression(C=0.5,solver="liblinear"),X_test,y_test, c
print(cv_results)
print('Logistic Regression cross validation score %f ' % cv_results.mean())
```

```
[0.9 0.9 0.8 0.9 0.8 1.  0.6 1.  0.8 0.8]
Logistic Regression cross validation score 0.850000
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

✓ 0s completed at 2:50 PM