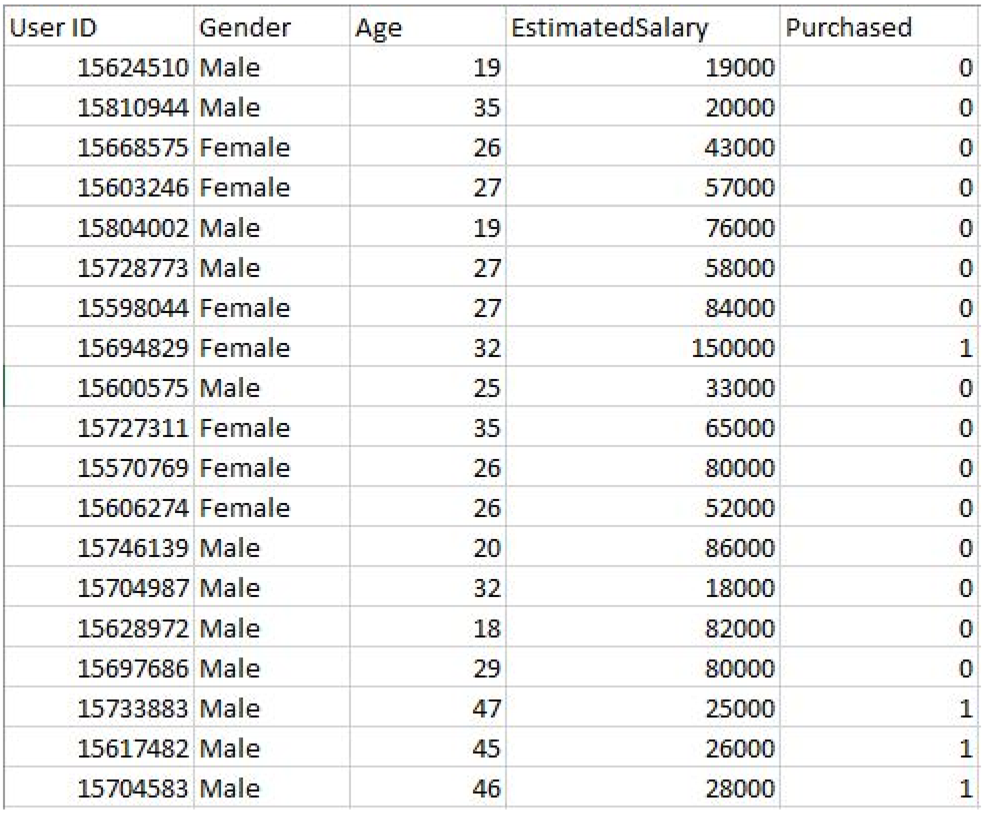
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
dataset = pd.read_csv("User_Data.csv")
```

User Database – This dataset contains information of users from a companies database. It contains information about UserID, Gender, Age, EstimatedSalary, Purchased. We are using this dataset for predicting that a user will purchase the company's newly launched product or not.

Data – User_Data



User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1

Let's make the Logistic Regression model, predicting whether a user will purchase the product or not.
Inputing Libraries

- Python3

```
import pandas as pd

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

Loading dataset – User_Data

```
dataset = pd.read_csv("User_Data.csv")
```

Now, to predict whether a user will purchase the product or not, one needs to find out the relationship between Age and Estimated Salary. Here User ID and Gender are not important factors for finding out this.

- Python3

```
# input

x = dataset.iloc[:, [2, 3]].values



# output

y = dataset.iloc[:, 4].values
```

Splitting the dataset to train and test. 75% of data is used for training the model and 25% of it is used to test the performance of our model.
Now, let's split our dataset into two: one to train our model and another to test our model. To do this, we use train_test_split imported from sklearn. Using a Logistic Regression Model, we will perform Classification on our train data and predict our test data to check the accuracy.

- Python3

```
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(

        x, y, test_size = 0.25, random_state = 0)
```

As we can see, our data contains a massive range of values, some are single digits, and some have three numbers. To make our calculations more straightforward, we will scale our data and reduce it to a small range of values using the Standard Scaler.

Now, it is very important to perform feature scaling here because Age and Estimated Salary values lie in different ranges. If we don’t scale the features then Estimated Salary feature will dominate Age feature when the model finds the nearest neighbor to a data point in data space.

- Python3

```
from sklearn.preprocessing import StandardScaler

sc_x = StandardScaler()

xtrain = sc_x.fit_transform(xtrain)

xtest = sc_x.transform(xtest)



print (xtrain[0:10, :])
```

Output :

```
[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]
 [-0.60673761  1.89663484]
 [ 1.37390747 -1.40858358]
 [ 1.47293972  0.99784738]
 [ 0.08648817 -0.79972756]
 [-0.01254409 -0.24885782]
 [-0.21060859 -0.5677824 ]
 [-0.21060859 -0.19087153]]
```

Here once see that Age and Estimated salary features values are sacled and now there in the -1 to 1. Hence, each feature will contribute equally in decision making i.e. finalizing the hypothesis.
Finally, we are training our Logistic Regression model.

- Python3

```
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state = 0)

classifier.fit(xtrain, ytrain)
```

After training the model, it time to use it to do prediction on testing data.

- Python3

```
y_pred = classifier.predict(xtest)
```

Let’s test the performance of our model – Confusion Matrix

- Python3

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(ytest, y_pred)



print ("Confusion Matrix : \n", cm)
```

Output :

```
Confusion Matrix :
 [[65  3]
 [ 8 24]]
```

Out of 100 :
TruePostive + TrueNegative = 65 + 24

FalsePositive + FalseNegative = 3 + 8

Performance measure – Accuracy

To find the accuracy of a confusion matrix and all other metrics, we can import accuracy_score and classification_report from the same library

- Python3

```
from sklearn.metrics import accuracy_score

print ("Accuracy : ", accuracy_score(ytest, y_pred))
```

Output :

```
Accuracy :  0.89
```

To Display Confusion Matrix

Using the predicted values(y_pred) and our actual values(ytest), we can create a confusion matrix with the confusion_matrix function.

Then, using the ravel() method of our confusion_matrix function, we can get the True Positive, True Negative, False Positive, and False Negative values.

```
from sklearn.metrics import confusion_matrix
tn=confusion_matrix(ytest,y_pred).ravel()
tn
```

References:-

https://www.simplilearn.com/tutorials/machine-learning-tutorial/confusion-matrix-machine-learning

https://www.geeksforgeeks.org/ml-logistic-regression-using-python/