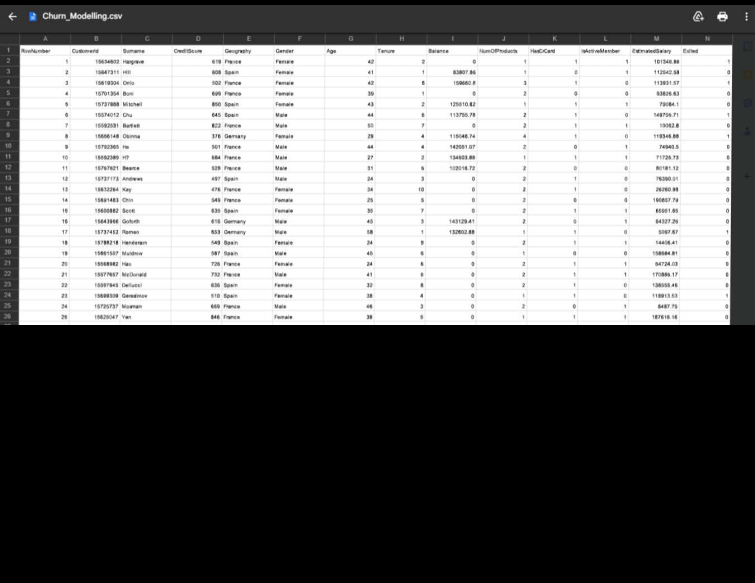

Churn_Modelling.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
2	1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.88	1
3	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
4	3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.57	1
5	4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
6	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.1	0
7	6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	149756.71	1
8	7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
9	8	15656148	Obinna	376	Germany	Female	29	4	115046.74	4	1	0	119346.88	1
10	9	15792365	He	501	France	Male	44	4	142051.07	2	0	1	74940.5	0
11	10	15592389	H?	684	France	Male	27	2	134603.88	1	1	1	71725.73	0
12	11	15767821	Bearce	528	France	Male	31	6	102016.72	2	0	0	80181.12	0
13	12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
14	13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
15	14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190857.79	0
16	15	15600882	Scott	635	Spain	Female	35	7	0	2	1	1	65951.65	0
17	16	15643966	Goforth	616	Germany	Male	45	3	143129.41	2	0	1	64327.26	0
18	17	15737452	Romeo	653	Germany	Male	58	1	132602.88	1	1	0	5097.67	1
19	18	15788218	Henderson	549	Spain	Female	24	9	0	2	1	1	14406.41	0
20	19	15661507	Muldrow	587	Spain	Male	45	6	0	1	0	0	158684.81	0
21	20	15568982	Hao	726	France	Female	24	6	0	2	1	1	54724.03	0
22	21	15577657	McDonald	732	France	Male	41	8	0	2	1	1	170886.17	0
23	22	15597945	Dellucci	636	Spain	Female	32	8	0	2	1	0	138555.46	0
24	23	15698538	Gerasimov	510	Spain	Female	38	4	0	1	1	0	118913.53	1
25	24	15735737	Mosman	669	France	Male	46	3	0	2	0	1	8487.75	0
26	25	15625047	Yan	846	France	Female	38	5	0	1	1	1	187616.16	0

Untitled7.ipynb

File Edit View Insert Runtime Tools Help Saving...

Comment Share

+ Code + Text

RAM Disk

Importing the libraries

```
[5] import numpy as np
    import pandas as pd
    import tensorflow as tf
```

```
[6] tf.__version__
```

```
'2.12.0'
```

Importing the dataset

Double-click (or enter) to edit

```
[7] dataset = pd.read_csv('Churn_Modelling.csv')
    x = dataset.iloc[:, 3:-1].values
    y = dataset.iloc[:, -1].values
```

```
[8] print(x)
```

Untitled7.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

```
print(y)
```

```
[1 0 1 ... 1 1 0]
```

+ Code + Text

Encoding the categorical data

```
[10] from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     x[:, 2] = le.fit_transform(x[:, 2])
```

```
[11] print(x)
```

```
[[619 'France' 0 ... 1 1 101348.88]
 [608 'Spain' 0 ... 0 1 112542.58]
 [502 'France' 0 ... 1 0 113931.57]
 ...
 [709 'France' 0 ... 0 1 42085.58]
 [772 'Germany' 1 ... 1 0 92888.52]
 [792 'France' 0 ... 1 0 38190.78]]
```

Untitled7.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

One hot encoding

```
[12] from sklearn.compose import ColumnTransformer
     from sklearn.preprocessing import OneHotEncoder
     ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder='passthrough')
     x= np.array(ct.fit_transform(x))
```

```
[13] print(x)
```

```
[[1.0 0.0 0.0 ... 1 1 101348.88]
 [0.0 0.0 1.0 ... 0 1 112542.58]
 [1.0 0.0 0.0 ... 1 0 113931.57]
 ...
 [1.0 0.0 0.0 ... 0 1 42085.58]
 [0.0 1.0 0.0 ... 1 0 92888.52]
 [1.0 0.0 0.0 ... 1 0 38190.78]]
```

Splitting into training set and test set

```
[14] from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)
```

Untitled7.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

Feature scaling

```
[15] from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     X_train = sc.fit_transform(X_train)
     X_test = sc.transform(X_test)
```

```
[ ]
```

Part-2 Building the ANN

```
[16] ann = tf.keras.models.Sequential()
```

```
[17] ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

```
[18] ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

```
[19] ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

Untitled7.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

Part-3 Training the ANN

```
[20] ann.compile(optimizer= 'adam',loss ='binary_crossentropy',metrics =['accuracy'])
```

```
[21] ann.fit(X_train, y_train, batch_size = 32, epochs = 100)
```

```
Epoch 1/100
250/250 [==============================] - 2s 2ms/step - loss: 0.6099 - accuracy: 0.6914
Epoch 2/100
250/250 [==============================] - 0s 2ms/step - loss: 0.4746 - accuracy: 0.7956
Epoch 3/100
250/250 [==============================] - 0s 2ms/step - loss: 0.4497 - accuracy: 0.7997
Epoch 4/100
250/250 [==============================] - 0s 2ms/step - loss: 0.4372 - accuracy: 0.8052
Epoch 5/100
250/250 [==============================] - 1s 2ms/step - loss: 0.4285 - accuracy: 0.8144
Epoch 6/100
250/250 [==============================] - 1s 3ms/step - loss: 0.4220 - accuracy: 0.8223
Epoch 7/100
250/250 [==============================] - 1s 2ms/step - loss: 0.4172 - accuracy: 0.8267
Epoch 8/100
250/250 [==============================] - 1s 3ms/step - loss: 0.4137 - accuracy: 0.8265
Epoch 9/100
250/250 [==============================] - 1s 3ms/step - loss: 0.4116 - accuracy: 0.8294
Epoch 10/100
250/250 [==============================] - 1s 3ms/step - loss: 0.4096 - accuracy: 0.8307
Epoch 11/100
250/250 [==============================] - 1s 3ms/step - loss: 0.4081 - accuracy: 0.8305
```

Untitled7.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Reconnect

Predicting the Test set result

```
[ ] y_pred = ann.predict(X_test)
    y_pred = (y_pred > 0.5)
    print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
63/63 [==============================] - 0s 1ms/step
[[0 0]
 [0 1]
 [0 0]
 ...
 [0 0]
 [0 0]
 [0 0]]
```

Making the confusion matrix

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

```
[[1522   73]
 [ 201  204]]
0.863
```