In [1]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import confusion_matrix, classification_report
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
```

In [2]:
```
churn_dataset = pd.read_csv("churn-bigml.csv")
```

In [3]:
```
churn_dataset.head()
```

Out[3]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	c
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99	
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103	
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	110	
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9	88	
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3	122	

In [4]:
```
churn_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   State                   3333 non-null   object
 1   Account length          3333 non-null   int64
 2   Area code               3333 non-null   int64
 3   International plan      3333 non-null   object
 4   Voice mail plan         3333 non-null   object
 5   Number vmail messages   3333 non-null   int64
 6   Total day minutes       3333 non-null   float64
 7   Total day calls         3333 non-null   int64
 8   Total day charge        3333 non-null   float64
 9   Total eve minutes       3333 non-null   float64
 10  Total eve calls         3333 non-null   int64
 11  Total eve charge        3333 non-null   float64
 12  Total night minutes     3333 non-null   float64
 13  Total night calls       3333 non-null   int64
 14  Total night charge      3333 non-null   float64
 15  Total intl minutes      3333 non-null   float64
 16  Total intl calls        3333 non-null   int64
 17  Total intl charge       3333 non-null   float64
 18  Customer service calls  3333 non-null   int64
 19  Churn                   3333 non-null   bool
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 498.1+ KB
```

In [5]:
```
X = churn_dataset.iloc[:,1:19]
```

```
Y = churn_dataset.iloc[:,19]
```

In [6]:
```
X = pd.get_dummies(X, columns=['International plan', 'Voice mail plan'], drop_first=
```

In [7]:
```
X.head()
```

Out[7]:

	Account length	Area code	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	To nig char
0	128	415	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11
1	107	415	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11
2	137	415	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7
3	84	408	0	299.4	71	50.90	61.9	88	5.26	196.9	89	8
4	75	415	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8

In [8]:
```
Y = pd.get_dummies(Y, columns=['CHurn'], drop_first=True)
```

In [9]:
```
Y.head()
```

Out[9]:

	True
0	0
1	0
2	0
3	0
4	0

In [10]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)
```

In [11]:
```
sc = StandardScaler()
```

In [12]:
```
X_train = sc.fit_transform(X_train)
```

In [13]:
```
X_train
```

Out[13]:
```
array([[-0.23658926, -0.69975651, -0.57936737, ..., -1.18386232,
        -0.32911022, -0.60623954],
       [ 0.92554395,  1.68630209, -0.57936737, ..., -0.42173462,
         3.0384957 , -0.60623954],
       [-1.27240364, -0.53600739, -0.57936737, ..., -0.42173462,
        -0.32911022, -0.60623954],
       ...,
       [ 0.42026864, -0.53600739, -0.57936737, ..., -1.18386232,
        -0.32911022, -0.60623954],
```

```
       [ 0.11710345, -0.53600739, -0.57936737, ..., -1.18386232,
        -0.32911022, -0.60623954],
       [-1.17134858, -0.53600739, -0.57936737, ..., -1.18386232,
        -0.32911022, -0.60623954]])
```

In [14]:
```
X_test = sc.transform(X_test)
```

In [15]:
```
def build_classifier(optimizer='adam'):
    clf = Sequential()
    clf.add(Dense(units=6, kernel_initializer='uniform', activation='relu', input_di
    clf.add(Dense(units=6, kernel_initializer='uniform', activation='relu'))
    clf.add(Dense(units=1, kernel_initializer='uniform', activation='sigmoid'))
    clf.compile(optimizer, loss='binary_crossentropy', metrics=['accuracy'])
    return clf
```

In [16]:
```
clf = KerasClassifier(build_fn=build_classifier, batch_size=10, epochs=100)
```

In [17]:
```
accuracies = cross_val_score(clf, X_train, Y_train, cv=10, n_jobs=-1)
```

In [18]:
```
mean = accuracies.mean()
std = accuracies.std()
print(f'Mean: {mean}')
print(f'Variance: {std*std}')
```

```
Mean: 0.8816844463348389
Variance: 0.0009073571593050644
```

In [19]:
```
accuracies
```

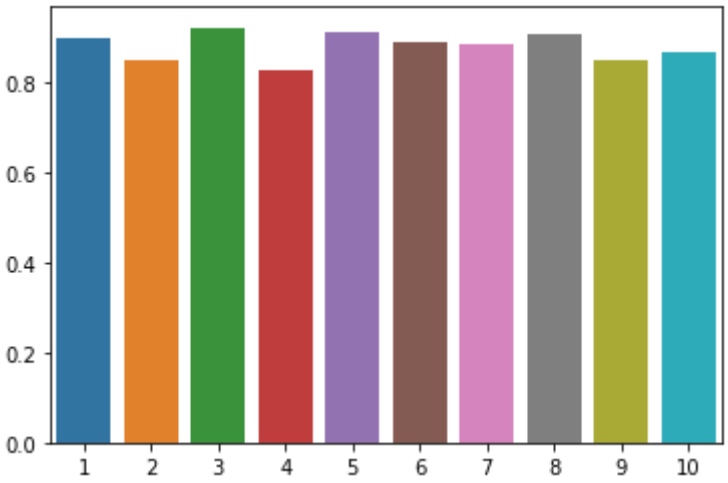
Out[19]:
```
array([0.90170938, 0.85042733, 0.92307693, 0.82832617, 0.91416311,
       0.888412  , 0.88412017, 0.90987122, 0.84978539, 0.86695278])
```

In [20]:
```
import seaborn as sns
```

In [21]:
```
x = [i for i in range(1, len(accuracies)+1)]
```

In [22]:
```
sns.barplot(x=x, y=accuracies)
```

Out[22]:
```
<AxesSubplot:>
```


0.0
0.2
0.4
0.6
0.8
1
2
3
4
5
6
7
8
9
10