In [74]:
```
#import pandas
import pandas as pd
#import numpy
import numpy as np
#import matplotlib
import matplotlib.pyplot as plt
#import seaborn
import seaborn as sns
```

In [75]:
```
# use pandas to import csv file
df = pd.read_csv("churn3.csv")
# too see max columns
pd.set_option('display.max_columns',None)
# print dataframe
df
```

Out[75]:

	Unnamed: 0	customer_id	vintage	age	gender	dependents	occupation	city	custome
0	0	1	2101	66	1.0	0.0	1.0	187.0	
1	1	2	2348	35	1.0	0.0	1.0	NaN	
2	2	4	2194	31	1.0	0.0	0.0	146.0	
3	3	5	2329	90	NaN	NaN	1.0	1020.0	
4	4	6	1579	42	1.0	2.0	1.0	1494.0	
...	...	...	...	...	...	...	...	...	
28377	28377	30297	2325	10	0.0	0.0	2.0	1020.0	
28378	28378	30298	1537	34	0.0	0.0	1.0	1046.0	
28379	28379	30299	2376	47	1.0	0.0	0.0	1096.0	
28380	28380	30300	1745	50	1.0	3.0	1.0	1219.0	
28381	28381	30301	1175	18	1.0	0.0	2.0	1232.0	

28382 rows × 11 columns

In [76]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28382 entries, 0 to 28381
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Unnamed: 0            28382 non-null  int64  
 1   customer_id           28382 non-null  int64  
 2   vintage               28382 non-null  int64  
 3   age                   28382 non-null  int64  
 4   gender                27857 non-null  float64
 5   dependents            25919 non-null  float64
 6   occupation            28302 non-null  float64
 7   city                  27579 non-null  float64
 8   customer_nw_category  28382 non-null  int64  
 9   branch_code           28382 non-null  int64  
 10  churn                 28382 non-null  int64  
dtypes: float64(4), int64(7)
memory usage: 2.4 MB
```

In [77]:
```
#New dataframe
new_df = df
#Checking for null values
print(new_df.isnull().sum())
print("Missing values distribution: ")
print(new_df.isnull().mean())
```

```
Unnamed: 0                0
customer_id               0
vintage                   0
age                       0
gender                  525
dependents             2463
occupation               80
city                    803
customer_nw_category      0
branch_code               0
churn                     0
dtype: int64
Missing values distribution:
Unnamed: 0              0.000000
customer_id             0.000000
vintage                 0.000000
age                     0.000000
gender                  0.018498
dependents              0.086780
occupation              0.002819
city                    0.028293
customer_nw_category    0.000000
branch_code             0.000000
churn                   0.000000
dtype: float64
```

In [78]:
```
#Replacing string values with integer
new_df["gender"].replace({"Male":"1", "Female":"0"}, inplace = True)
print(new_df.head())
```

```
   Unnamed: 0  customer_id  vintage  age  gender  dependents  occupation  \
0           0            1     2101   66     1.0         0.0         1.0
1           1            2     2348   35     1.0         0.0         1.0
2           2            4     2194   31     1.0         0.0         0.0
3           3            5     2329   90     NaN         NaN         1.0
4           4            6     1579   42     1.0         2.0         1.0

     city  customer_nw_category  branch_code  churn
0   187.0                     2          755      0
1     NaN                     2         3214      0
2   146.0                     2           41      0
3  1020.0                     2          582      1
4  1494.0                     3          388      1
```

In [79]:
```
#Replacing string values with integer
new_df["occupation"].replace({"salaried":"0", "self_employed":"1","student":"2
print(new_df.head())
```

```
   Unnamed: 0  customer_id  vintage  age  gender  dependents  occupation  \
0           0            1     2101   66     1.0         0.0         1.0
1           1            2     2348   35     1.0         0.0         1.0
2           2            4     2194   31     1.0         0.0         0.0
3           3            5     2329   90     NaN         NaN         1.0
4           4            6     1579   42     1.0         2.0         1.0

     city  customer_nw_category  branch_code  churn
0   187.0                     2          755      0
1     NaN                     2         3214      0
2   146.0                     2           41      0
3  1020.0                     2          582      1
4  1494.0                     3          388      1
```

In [80]:
```
#drop missing values
df2= new_df.dropna(axis=1)
```

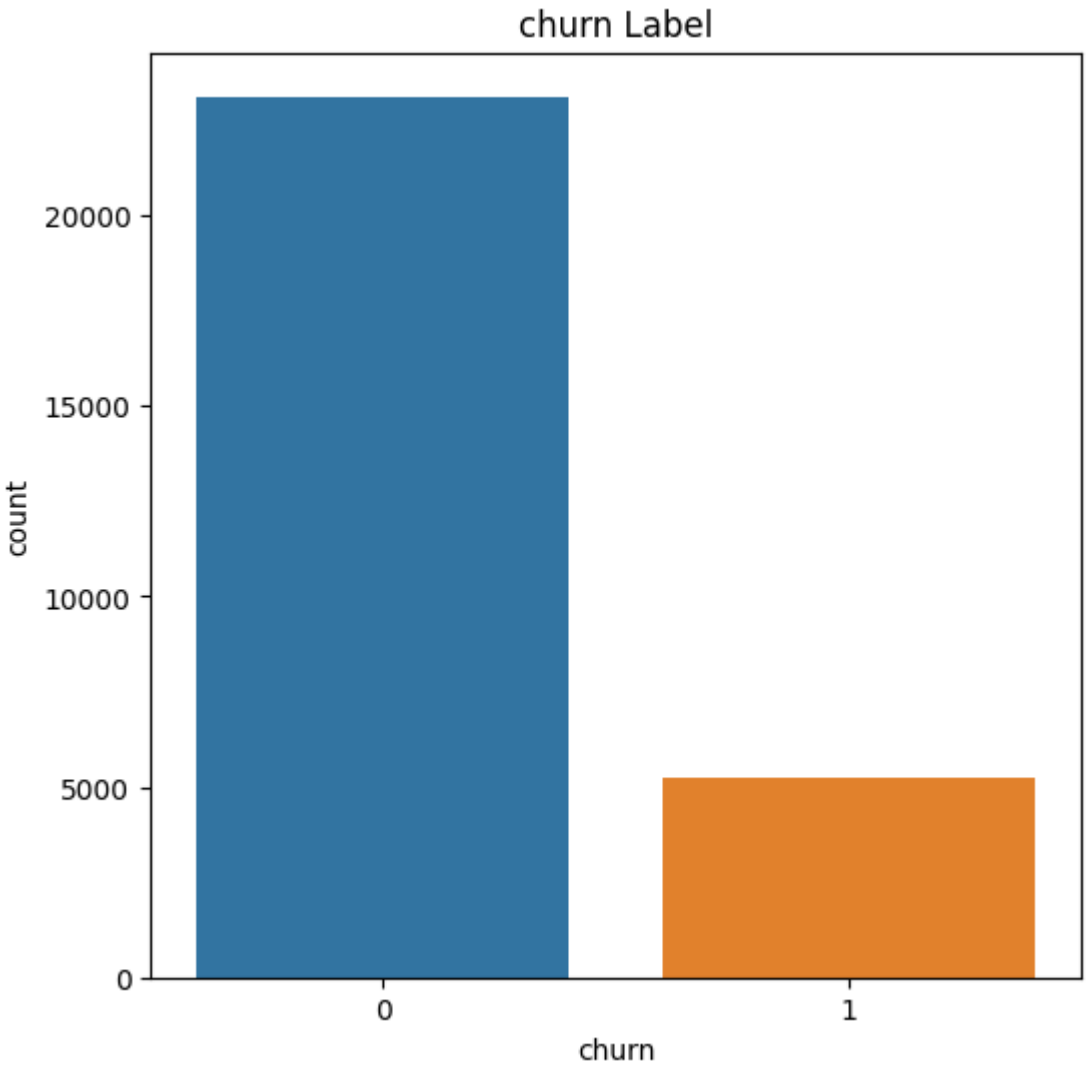
In [82]:
```
df2.head
```

Out[82]:
```
<bound method NDFrame.head of       Unnamed: 0  customer_id  vintage  age  c
ustomer_nw_category  \
0                0            1     2101   66                    2
1                1            2     2348   35                    2
2                2            4     2194   31                    2
3                3            5     2329   90                    2
4                4            6     1579   42                    3
...            ...          ...      ...  ...                  ...
28377        28377        30297     2325   10                    2
28378        28378        30298     1537   34                    2
28379        28379        30299     2376   47                    2
28380        28380        30300     1745   50                    3
28381        28381        30301     1175   18                    2

       branch_code  churn
0              755      0
1             3214      0
2               41      0
3              582      1
4              388      1
...            ...    ...
28377         1207      0
28378          223      0
28379          588      1
28380          274      0
28381          474      1

[28382 rows x 7 columns]>
```

In [87]:
```
df2.churn.value_counts()
plt.figure(figsize=(6,6))
sns.countplot(x='churn', data=new_df)
plt.title('churn Label')
plt.show()
```



In [88]:
```
from sklearn.datasets import make_classification
X, y = make_classification(n_classes=2, class_sep=0.5,
weights=[0.05, 0.95], n_informative=2, n_redundant=0, flip_y=0,
n_features=2, n_clusters_per_class=1, n_samples=1000, random_state=10)
```

In [89]:
```
from sklearn.model_selection import train_test_split

# split into 75:25 ratio
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, ra
```

In [90]:
```
#KNN classifer
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
#Accuracy and Confusion matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(accuracy_score(y_test, model.predict(X_test)))
cm
```

```
0.98
```

Out[90]:
```
array([[ 10,   5],
       [  0, 235]], dtype=int64)
```

In [83]:
```
#Naive Bayes Classifier
x = df2.drop(["churn"], axis = 1)
y = df2.churn.values
from sklearn.model_selection import train_test_split

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, ran
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(x_train, y_train)
print("Naive Bayes score: ",nb.score(x_test, y_test))
```

```
Naive Bayes score:  0.8182031708749266
```

In [99]:
```
#split dataset in features and target variable
feature_cols = ['age', 'customer_id', 'customer_nw_category']
X = df2[feature_cols] # Features
y = df2.churn # Target variable
```

In [100]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, randc
```

In [101]:
```
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classif
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

In [102]:
```
from sklearn import metrics #Import scikit-learn metrics module for accuracy c
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.6903112155020552
```

In [ ]:

In [ ]: