In [8]:

```
# Imports
%matplotlib inline
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn_evaluation import plot
import category_encoders as ce
import matplotlib.pyplot as plt
import seaborn as sns

# Setup Seaborn
sns.set_style("whitegrid")
sns.set_context("poster")
```

In [9]:

```
# Read CSV from https://www.kaggle.com/datasets/shubh0799/churn-modelling
df_csv = pd.read_csv('Churn_Modeling.csv')
df_csv.isna().describe()
# No N/A to deal with
```

Out[9]:

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balan |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 10000 | 10000 | 10000 | 10000 | 10000 | 10000 | 10000 | 10000 | 100 |
| **unique** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| **top** | False | False | False | False | False | False | False | False | Fal |
| **freq** | 10000 | 10000 | 10000 | 10000 | 10000 | 10000 | 10000 | 10000 | 100 |

In [10]:

```
df_csv.describe()
# Initial distributions of features
```

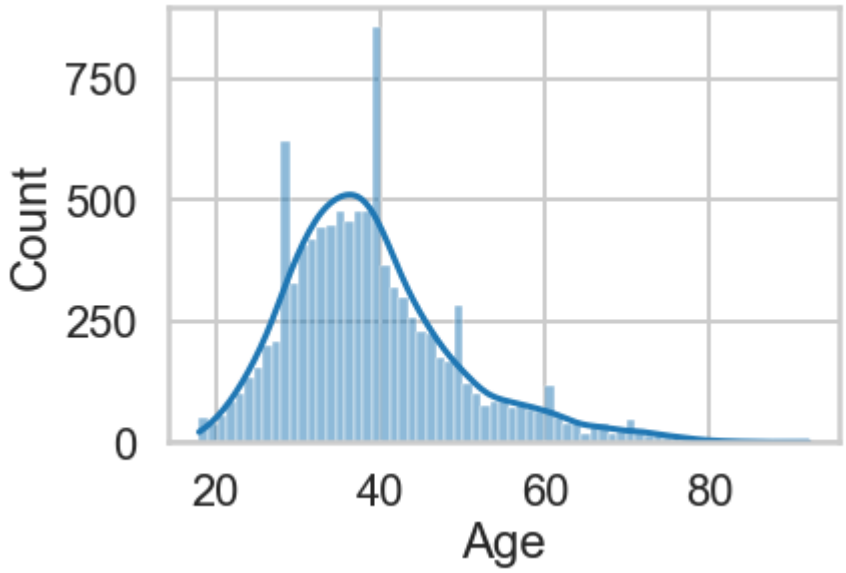
Out[10]:

| | RowNumber | CustomerId | CreditScore | Age | Tenure | Balance | NumOf |
|---|---|---|---|---|---|---|---|
| **count** | 10000.00000 | 1.000000e+04 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 100 |
| **mean** | 5000.50000 | 1.569094e+07 | 650.528800 | 38.921800 | 5.012800 | 76485.889288 | |
| **std** | 2886.89568 | 7.193619e+04 | 96.653299 | 10.487806 | 2.892174 | 62397.405202 | |
| **min** | 1.00000 | 1.556570e+07 | 350.000000 | 18.000000 | 0.000000 | 0.000000 | |
| **25%** | 2500.75000 | 1.562853e+07 | 584.000000 | 32.000000 | 3.000000 | 0.000000 | |
| **50%** | 5000.50000 | 1.569074e+07 | 652.000000 | 37.000000 | 5.000000 | 97198.540000 | |
| **75%** | 7500.25000 | 1.575323e+07 | 718.000000 | 44.000000 | 7.000000 | 127644.240000 | |

| | RowNumber | CustomerId | CreditScore | Age | Tenure | Balance | NumOf |
|---|---|---|---|---|---|---|---|
| **max** | 10000.00000 | 1.581569e+07 | 850.000000 | 92.000000 | 10.000000 | 250898.090000 | |

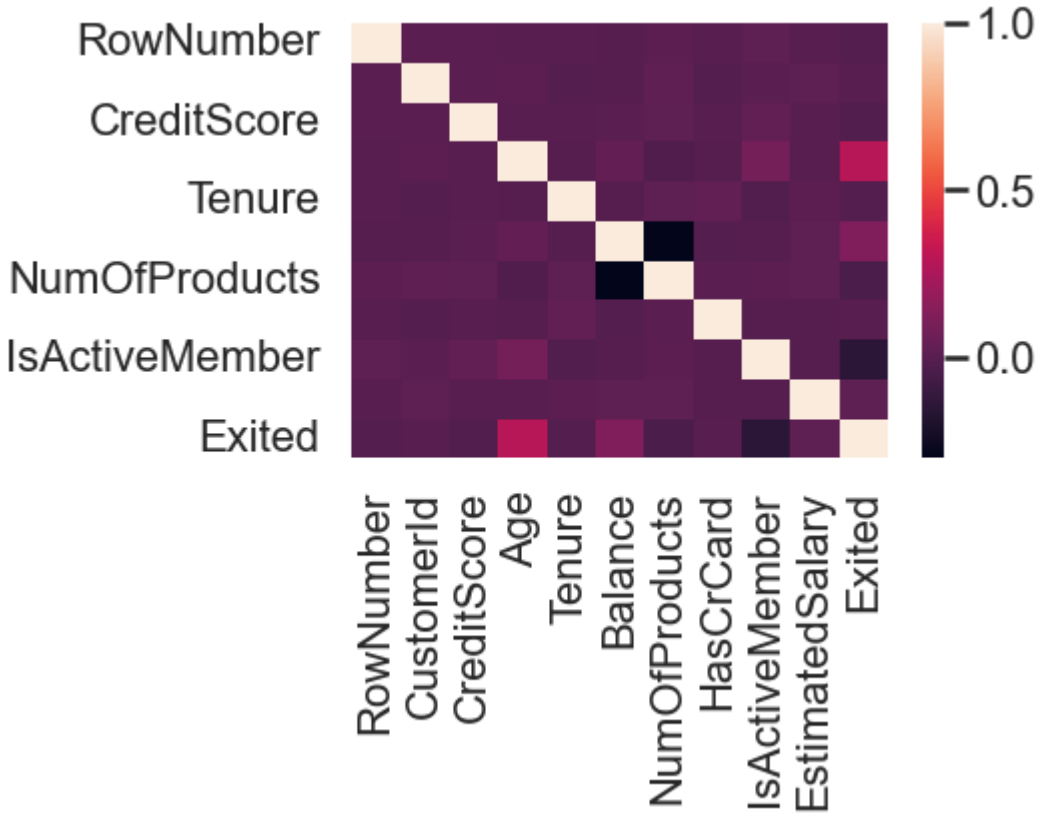
In [11]:
```
sns.histplot(data = df_csv['Age'], kde = True)
# Age median seen around spike at 40
```

Out[11]: <AxesSubplot:xlabel='Age', ylabel='Count'>




In [12]:
```
sns.heatmap(df_csv.corr())
# Heatmap of correlation show Age and Balance as highest corr to Exited
```

Out[12]: <AxesSubplot:>




In [13]:

```
print(len(df_csv['Surname'].unique()))
print(df_csv['Geography'].unique())
print(df_csv['Gender'].unique())
print(len(df_csv['CustomerId'].unique()))
print(df_csv.dtypes)

# We see that:
# Row number is simply ascending 1 to 10,000
# Surname is 2,932 unique values in 10,000 rows
# Geography has 3 unique countries represented
# Gender has 2 unique genders represented
# Customer ID is 10,000 unique values in 10,000 rows of the df
# Credit score is from 350 to 850, integers only
# Age is from 18 to 92, integers only
# Tenure is from 0 to 10, integers only
# Balance is from 0 to ~251,000, float value
# NumOfProducts is from 1 to 4, integers only
# HasCrCard is boolean
# IsActiveMember is boolean
# Estimated Salary is ~11.6 to ~200,000, float value
# Exited is boolean
```

```
2932
['France' 'Spain' 'Germany']
['Female' 'Male']
10000
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
```

```
Age                 int64
Tenure              int64
Balance           float64
NumOfProducts       int64
HasCrCard           int64
IsActiveMember      int64
EstimatedSalary   float64
Exited              int64
dtype: object
```

In [14]:

```
df_csv[df_csv['IsActiveMember']==df_csv['Exited']]
# There are cases (43% of data) of inactive members that have not exited or active m
# 'Exited' column will be used as the target variable
```

Out[14]:

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0.00 |
| **3** | 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | 0.00 |
| **10** | 11 | 15767821 | Bearce | 528 | France | Male | 31 | 6 | 102016.72 |
| **11** | 12 | 15737173 | Andrews | 497 | Spain | Male | 24 | 3 | 0.00 |
| **12** | 13 | 15632264 | Kay | 476 | France | Female | 34 | 10 | 0.00 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **9993** | 9994 | 15569266 | Rahman | 644 | France | Male | 28 | 7 | 155060.41 |
| **9994** | 9995 | 15719294 | Wood | 800 | France | Female | 29 | 2 | 0.00 |
| **9995** | 9996 | 15606229 | Obijiaku | 771 | France | Male | 39 | 5 | 0.00 |
| **9997** | 9998 | 15584532 | Liu | 709 | France | Female | 36 | 7 | 0.00 |
| **9999** | 10000 | 15628319 | Walker | 792 | France | Female | 28 | 4 | 130142.79 |

4282 rows × 14 columns

In [15]:

```
df_csv['Exited'].value_counts()
# ~80% of customers have not exited, we will need to deal with class imbalance
```

Out[15]:

```
0    7963
1    2037
Name: Exited, dtype: int64
```

In [16]:

```
## One hot encode Geography and Gender, scale float features to 0-1
# RowNumber, drop
# Surname, drop
# Geography, one hot
# Gender, one hot
# Customer ID, index
# Credit score, scale
# Age, scale
# Tenure, scale
# Balance, scale
# NumOfProducts scale
```

```
# HasCrCard, none
# IsActiveMember none
# EstimatedSalary, scale
# Exited, target

oh_encode = ce.one_hot.OneHotEncoder(cols=
    ['Geography', 'Gender'], use_cat_names=True)
df_encode= oh_encode.fit_transform(df_csv)

scaler = MinMaxScaler()
df_encode[['CreditScore_Scaled', 'Age_Scaled', 'Tenure_Scaled', 'Balance_Scaled', 'N
    scaler.fit_transform(df_encode[['CreditScore', 'Age', 'Tenure', 'Balance', 'NumC

df_encode.set_index('CustomerId', inplace=True)
df_encode.drop(['RowNumber', 'Surname', 'CreditScore', 'Age', 'Tenure', 'Balance', '

df_encode.head()
```

Out[16]:

| CustomerId | Geography_France | Geography_Spain | Geography_Germany | Gender_Female | Gender_Male |
|---|---|---|---|---|---|
| **15634602** | 1 | 0 | 0 | 1 | 0 |
| **15647311** | 0 | 1 | 0 | 1 | 0 |
| **15619304** | 1 | 0 | 0 | 1 | 0 |
| **15701354** | 1 | 0 | 0 | 1 | 0 |
| **15737888** | 0 | 1 | 0 | 1 | 0 |

In [17]:

```
# Train test split at 70%
X = df_encode[['CreditScore_Scaled', 'Age_Scaled', 'NumOfProducts_Scaled', 'Balance_
       'EstimatedSalary_Scaled', 'Tenure_Scaled', 'HasCrCard', 'IsActiveMember',
       'Gender_Female', 'Gender_Male',
       'Geography_France', 'Geography_Spain', 'Geography_Germany',]]

y = df_encode['Exited']

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, shuffle=Tr
```

In [18]:

```
# Target encoding for boolean features
tar_enc = ce.TargetEncoder(cols=['HasCrCard', 'IsActiveMember',
       'Geography_France', 'Geography_Spain', 'Geography_Germany',
       'Gender_Female', 'Gender_Male'],
       min_samples_leaf=5, smoothing=3).fit(X_train, y_train)
X_train = tar_enc.transform(X_train)
X_test = tar_enc.transform(X_test)
```

In [19]:

```
# Grid Search from: https://stackoverflow.com/questions/38151615/specific-cross-vali
clf = RandomForestClassifier()
param_grid = {'n_estimators': [50,100,150,200,250], 'max_depth': [10,None], 'max_fea
    'min_samples_leaf': [1,3,6,10]}
```

```
grid_clf = GridSearchCV(clf, param_grid, cv=5)
grid_clf.fit(X_train, y_train)

print(grid_clf.best_estimator_)
```

```
RandomForestClassifier(max_depth=10, max_features=5)
```

In [20]:

```
cv_res = pd.DataFrame(grid_clf.cv_results_)
cv_res[cv_res['rank_test_score']<=5].sort_values(by=['rank_test_score'])
```

Out[20]:

| | mean_fit_time | std_fit_time | mean_score_time | std_score_time | param_max_depth | param_max_fea |
|---|---|---|---|---|---|---|
| **21** | 0.752607 | 0.023861 | 0.029798 | 0.002029 | 10 | |
| **27** | 1.197115 | 0.073086 | 0.045817 | 0.005108 | 10 | |
| **42** | 1.533411 | 0.031623 | 0.041942 | 0.001705 | 10 | |
| **20** | 0.411551 | 0.020454 | 0.019301 | 0.002791 | 10 | |
| **131** | 0.779007 | 0.032427 | 0.028935 | 0.000458 | None | |

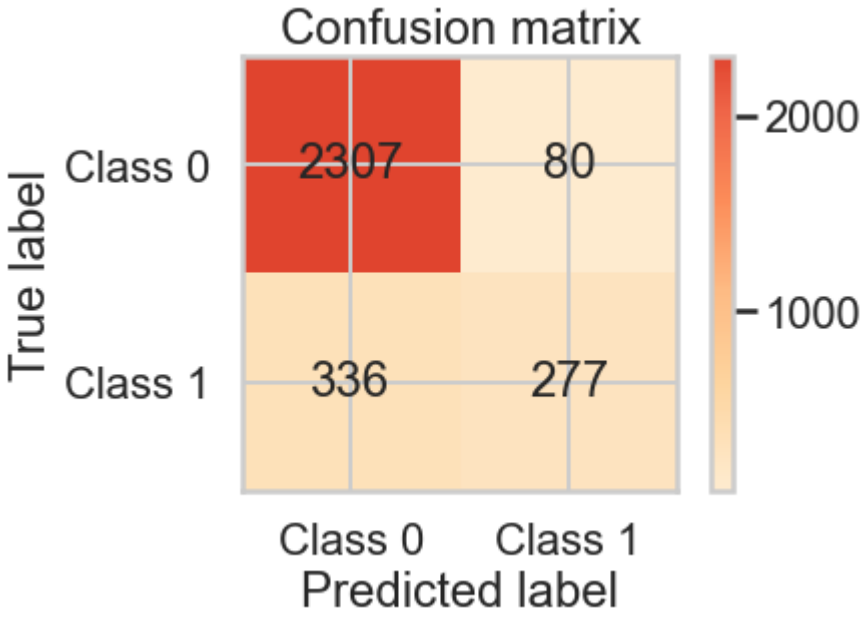
In [22]:

```
# Training the model on the training dataset
# fit function is used to train the model using the training sets as parameters
clf = RandomForestClassifier(max_depth = 10, min_samples_leaf=1,
    max_features=5, n_estimators=100)
clf.fit(X_train, y_train)

# performing predictions on the test dataset
y_pred = clf.predict(X_test)

# using metrics module for accuracy calculation
print('ACCURACY: ', metrics.accuracy_score(y_test, y_pred))
print('PRECISION: ', metrics.precision_score(y_test, y_pred))
print('RECALL: ', metrics.recall_score(y_test, y_pred))
print('F1: ', metrics.f1_score(y_test, y_pred))
plot.confusion_matrix(y_test, y_pred)
# ~80% of customers have not exited, try SMOTE or Tomek?
# Strong bias to unexited customers, 50/50 on exited customers
```

```
ACCURACY:  0.8613333333333333
PRECISION:  0.7759103641456583
```

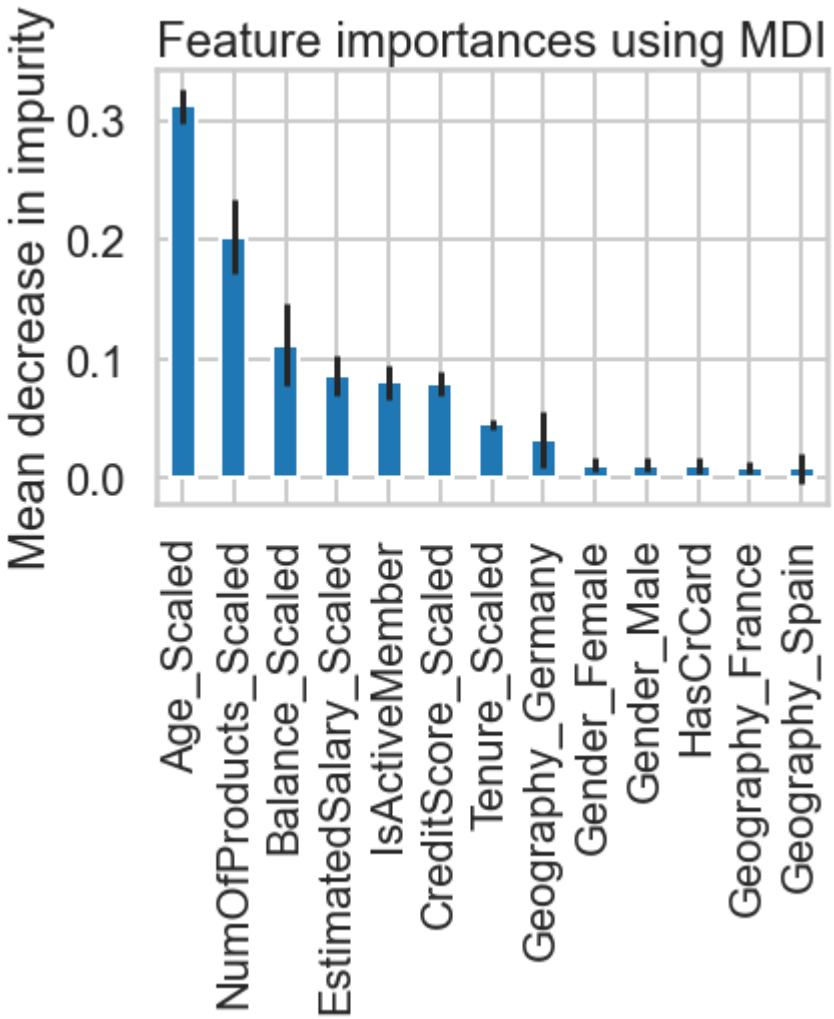
```
RECALL:  0.45187601957585644
F1:  0.5711340206185568
```

Out[22]:
```
<AxesSubplot:title={'center':'Confusion matrix'}, xlabel='Predicted label', ylabel
='True label'>
```




In [23]:
```
# Feature Importance, strongest predictors seen are: age, num products, balance, cre
# https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.htm
import matplotlib.pyplot as plt
importances = clf.feature_importances_
std = np.std([tree.feature_importances_ for tree in clf.estimators_], axis=0)
forest_importances = pd.Series(importances, index=X_train.columns).sort_values(ascer

fig, ax = plt.subplots()
forest_importances.plot.bar(yerr=std, ax=ax)
ax.set_title("Feature importances using MDI")
ax.set_ylabel("Mean decrease in impurity")
```

Out[23]:
```
Text(0, 0.5, 'Mean decrease in impurity')
```


Feature importances using MDI
Mean decrease in impurity
0.0
0.1
0.2
0.3
Age_Scaled
NumOfProducts_Scaled
Balance_Scaled
EstimatedSalary_Scaled
IsActiveMember
CreditScore_Scaled
Tenure_Scaled
Geography_Germany
Gender_Female
Gender_Male
HasCrCard
Geography_France
Geography_Spain