Bank Customer Churn Prediction

In this kernel I am going to make an **Exploratory Data Analysis (EDA)** on this (https://www.kaggle.com/filippoo/deep-learning-az-ann) dataset. Also I am going to make different predictive models and find out the best one with highest prediction accuracy.

Kernel Outlines:

- **Importing Necessary Packages**
- **Statistical Summary of the Dataset**
- **Dropping Irrelevant Features**
- **One Hot Encoding**
- **Data Visualization**
- **Detecting Outliers using Tukey Boxplot**
- **Hand written function for detecting and removing outliers**
- **Checking Correlation with Heatmap**
- **Different ML predictive models**
  - Gaussian Naive Bayes
  - Logistic Regression
  - Decision Tree
  - Random Forest
  - Extra Gradient Boosting Tree (XGBoost)
- **Improve the Predictive Model**
  - Feature Scaling
  - Over Sampling

Importing Necessary Packages

In [1]:

```
import numpy as np
import pandas as pd
import tensorflow as tf
import keras

import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
sns.set(palette="Set2")

from sklearn.model_selection import train_test_split
from sklearn.metrics import (accuracy_score, f1_score,average_precisio
n_score, confusion_matrix,
                             average_precision_score, precision_score,
recall_score, roc_auc_score, )
from mlxtend.plotting import plot_confusion_matrix
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn import tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder, MinMaxScaler, Standard
Scaler


from xgboost import XGBClassifier, plot_importance
from imblearn.over_sampling import SMOTE
```


In [2]:

```
# read dataset
dataset = pd.read_csv("../input/Churn_Modelling.csv")
```


In [3]:

```
# first five row of the dataset
dataset.head()
```

Out[3]:

r	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
e	42	2	0.00	1	1	1	101348.88	1
e	41	1	83807.86	1	0	1	112542.58	0
e	42	8	159660.80	3	1	0	113931.57	1
e	39	1	0.00	2	0	0	93826.63	0
e	43	2	125510.82	1	1	1	79084.10	0

The statistical summary of the dataset

In [4]:

```
dataset.describe()
```

Out[4]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000
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
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000

In [5]:

```
# checking datatypes and null values
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
RowNumber          10000 non-null int64
CustomerId         10000 non-null int64
Surname            10000 non-null object
```



```
Age                10000 non-null int64
Tenure             10000 non-null int64
Balance            10000 non-null float64
NumOfProducts      10000 non-null int64
HasCrCard          10000 non-null int64
IsActiveMember     10000 non-null int64
EstimatedSalary    10000 non-null float64
Exited             10000 non-null int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

Dropping Irrelevant Feature

`RowNumber`, `CustomerId` and `Surname` are irrelivant, so we drop those features.

In [6]:

```
dataset.drop(["RowNumber","CustomerId","Surname"], axis=1, inplace=Tru
e)
```

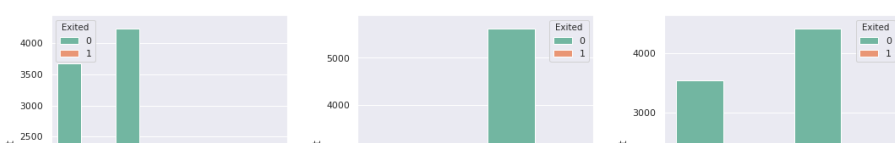
Data Visualization

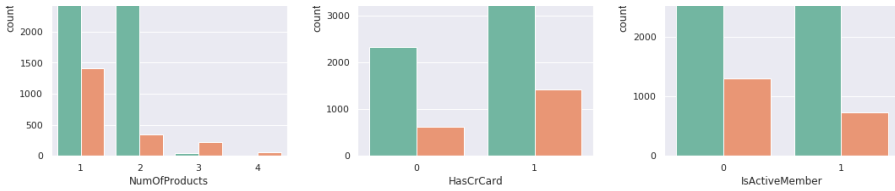
In [7]:

```
_, ax = plt.subplots(1, 3, figsize=(18, 6))
plt.subplots_adjust(wspace=0.3)
sns.countplot(x = "NumOfProducts", hue="Exited", data = dataset, ax= a
x[0])
sns.countplot(x = "HasCrCard", hue="Exited", data = dataset, ax = ax[1
])
sns.countplot(x = "IsActiveMember", hue="Exited", data = dataset, ax =
ax[2])
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff974d0fc18>
```





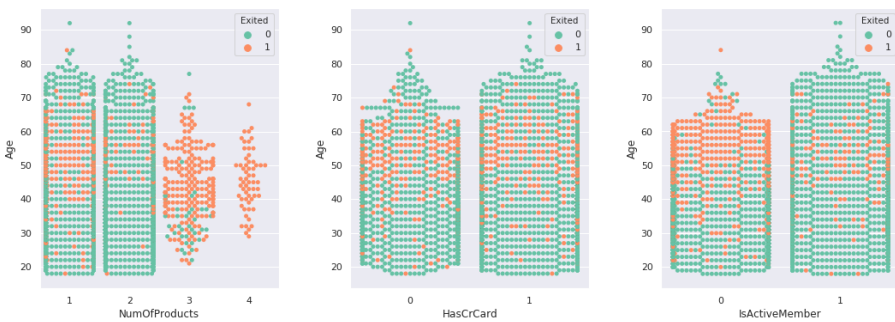
Customer with 3 or 4 products are higher chances to Churn

In [8]:

```
_, ax = plt.subplots(1, 3, figsize=(18, 6))
plt.subplots_adjust(wspace=0.3)
sns.swarmplot(x = "NumOfProducts", y = "Age", hue="Exited", data = dat
aset, ax= ax[0])
sns.swarmplot(x = "HasCrCard", y = "Age", data = dataset, hue="Exited"
, ax = ax[1])
sns.swarmplot(x = "IsActiveMember", y = "Age", hue="Exited", data = da
taset, ax = ax[2])
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff97436b160>
```



In [9]:

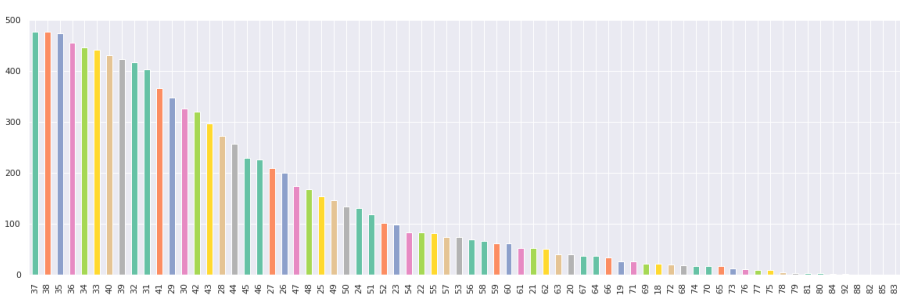
```
encoder = LabelEncoder()
dataset["Geography"] = encoder.fit_transform(dataset["Geography"])
dataset["Gender"] = encoder.fit_transform(dataset["Gender"])
```

In [10]:

```
dataset["Age"].value_counts().plot.bar(figsize=(20,6))
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff974c61dd8>
```



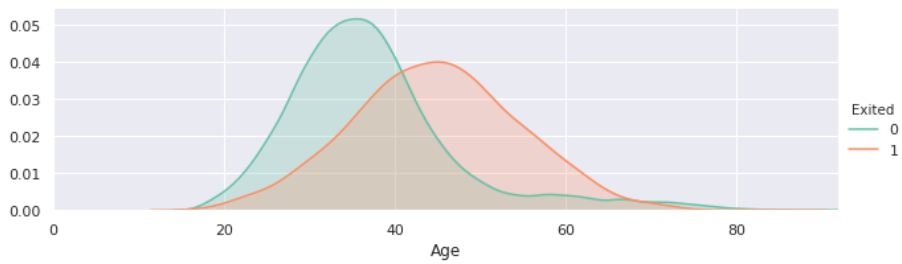
In [11]:

In [11]:

```
facet = sns.FacetGrid(dataset, hue="Exited",aspect=3)
facet.map(sns.kdeplot,"Age",shade= True)
facet.set(xlim=(0, dataset["Age"].max()))
facet.add_legend()

plt.show()
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: Futu
reWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future
this will be interpreted as an array index, `arr[np.array(seq)]`, whic
h will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

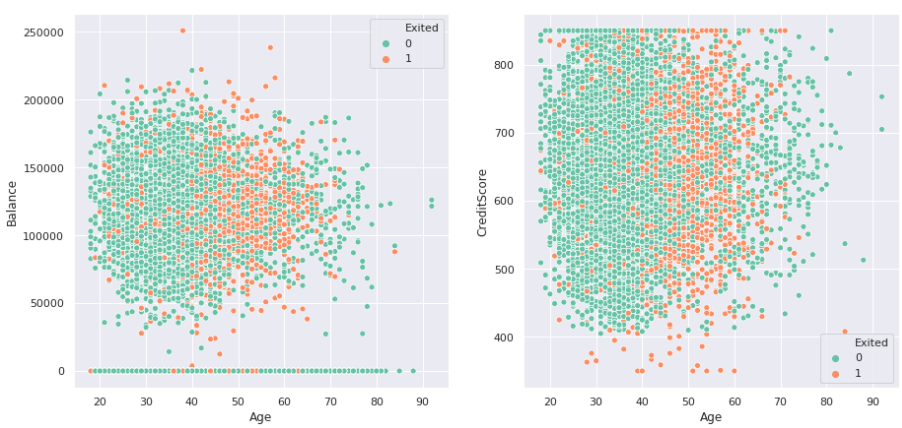


In [12]:

```
_, ax =  plt.subplots(1, 2, figsize=(15, 7))
cmap = sns.cubehelix_palette(light=1, as_cmap=True)
sns.scatterplot(x = "Age", y = "Balance", hue = "Exited", cmap = cmap,
sizes = (10, 200), data = dataset, ax=ax[0])
sns.scatterplot(x = "Age", y = "CreditScore", hue = "Exited", cmap = c
map, sizes = (10, 200), data = dataset, ax=ax[1])
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff96b7e3080>
```



- **40 to 70 years old customers are higher chances to churn**
- **Customer with `CreditScore` less then `400` are higher chances to churn**

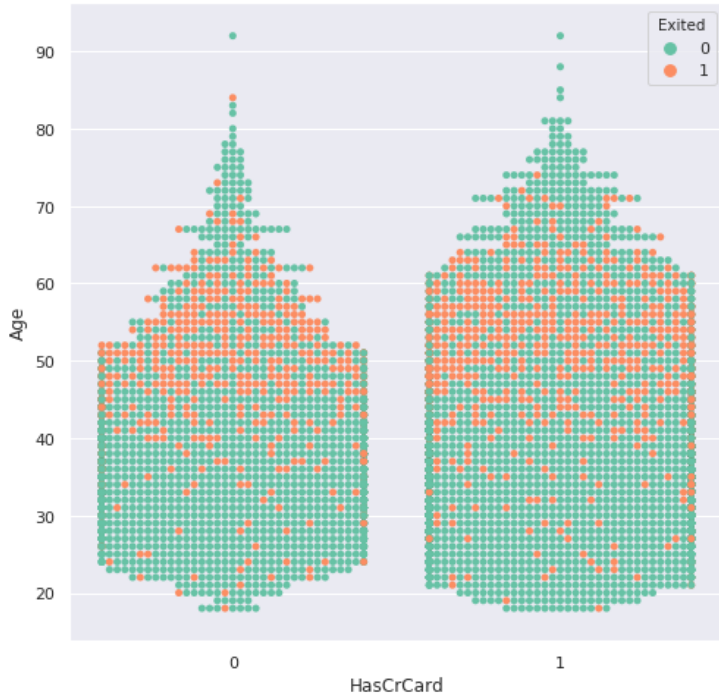
In [13]:

```
plt.figure(figsize=(8, 8))
```

```
sns.swarmplot(x = "HasCrCard", y = "Age", data = dataset, hue="Exited"
)
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff9742988d0>
```

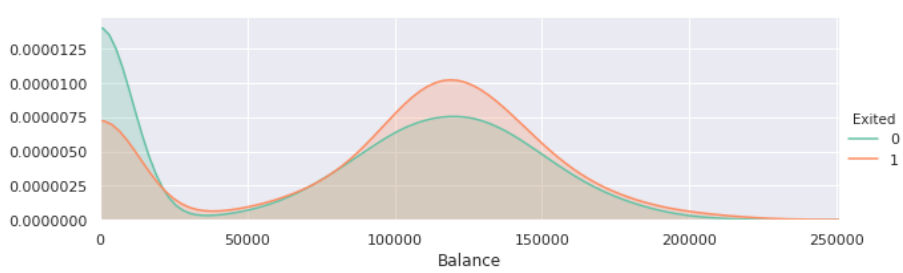


In [14]:

```
facet = sns.FacetGrid(dataset, hue="Exited",aspect=3)
facet.map(sns.kdeplot,"Balance",shade= True)
facet.set(xlim=(0, dataset["Balance"].max()))
facet.add_legend()

plt.show()
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: Futu
reWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future
this will be interpreted as an array index, `arr[np.array(seq)]`, whic
h will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



In [15]:

```
_, ax = plt.subplots(1, 2, figsize=(15, 6))
sns.scatterplot(x = "Balance", y = "Age", data = dataset, hue="Exited"
```

```
, ax = ax[0])
sns.scatterplot(x = "Balance", y = "CreditScore", data = dataset, hue=
"Exited", ax = ax[1])
```

Out[15]:

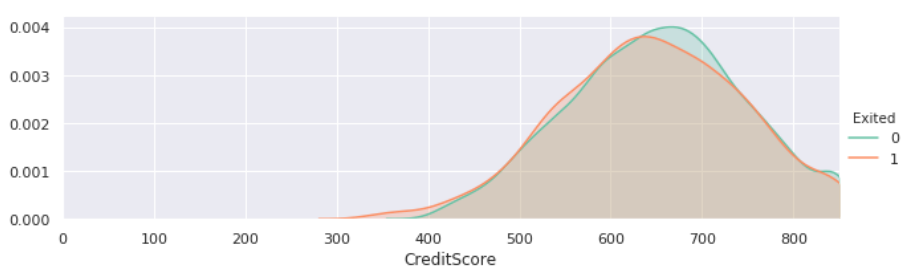
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff96b602b38>
```

In [16]:

```
facet = sns.FacetGrid(dataset, hue="Exited",aspect=3)
facet.map(sns.kdeplot,"CreditScore",shade= True)
facet.set(xlim=(0, dataset["CreditScore"].max()))
facet.add_legend()

plt.show()
```

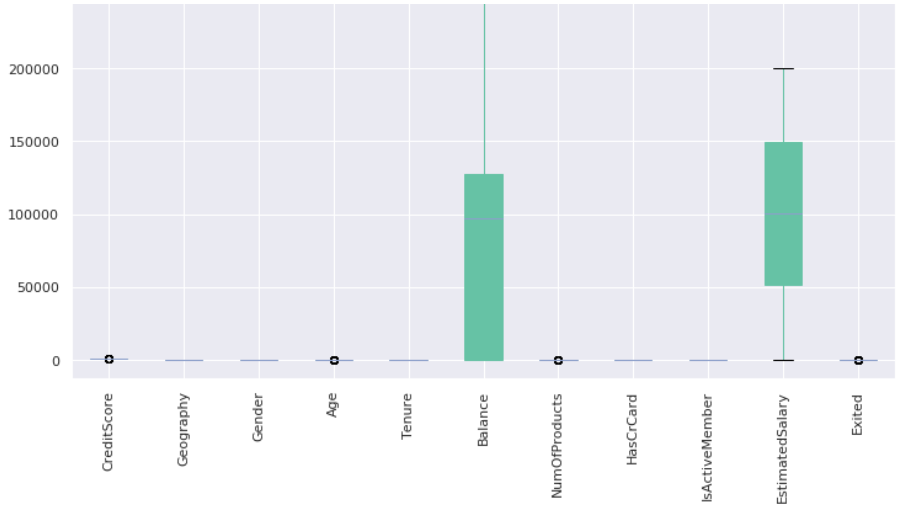
```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: Futu
reWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future
this will be interpreted as an array index, `arr[np.array(seq)]`, whic
h will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



Detecting Outliers using Tukey Boxplot

In [17]:

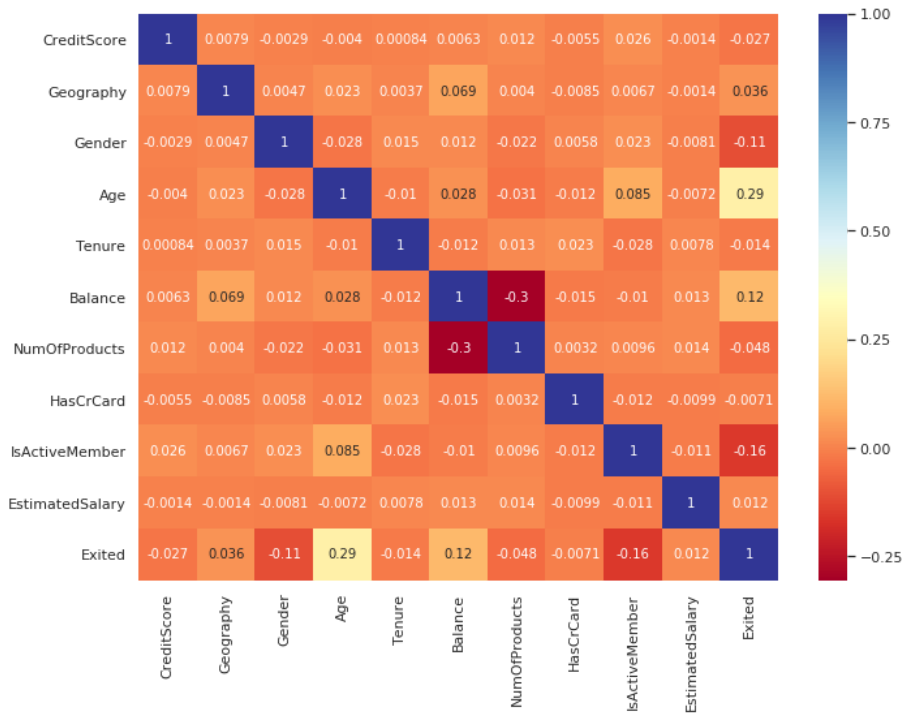
```
plt.figure(figsize=(12,6))
bplot = dataset.boxplot(patch_artist=True)
plt.xticks(rotation=90)
plt.show()
```



Checking Correlation

In [18]:

```
plt.subplots(figsize=(11,8))
sns.heatmap(dataset.corr(), annot=True, cmap="RdYlBu")
plt.show()
```



Prediction with ML models:

In [19]:

```
X = dataset.drop("Exited", axis=1)
```

```
y = dataset["Exited"]
```

In [20]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
2, random_state=0)
```

In [21]:

```
clf = GaussianNB()
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
accuracy_score(pred, y_test)
```

Out[21]:

```
0.784
```

In [22]:

```
clf = LogisticRegression()
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
accuracy_score(pred, y_test)
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.p
y:433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.2
2. Specify a solver to silence this warning.
  FutureWarning)
```

Out[22]:

```
0.787
```

In [23]:

```
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
accuracy_score(pred, y_test)
```

Out[23]:

```
0.7915
```

In [24]:

```
clf = RandomForestClassifier(n_estimators = 200, random_state=200)
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
accuracy_score(pred, y_test)
```

Out[24]:

```
0.864
```

In [25]:

```
clf  = XGBClassifier(max_depth = 10,random_state = 10, n_estimators=22
0, eval_metric = 'auc', min_child_weight = 3,
                    colsample_bytree = 0.75, subsample= 0.9)

clf.fit(X_train, y_train)
pred = clf.predict(X_test)
accuracy_score(pred, y_test)
```

Out[25]:

```
0.8575
```

In [26]:

```
scaler = MinMaxScaler()

bumpy_features = ["CreditScore", "Age", "Balance",'EstimatedSalary']

df_scaled = pd.DataFrame(data = X)
df_scaled[bumpy_features] = scaler.fit_transform(X[bumpy_features])
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/preprocessing/data.py:3
23: DataConversionWarning: Data with input dtype int64, float64 were a
ll converted to float64 by MinMaxScaler.
  return self.partial_fit(X, y)
```

In [27]:

```
df_scaled.head()
```

Out[27]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard
0	0.538	0	0	0.324324	2	0.000000	1	1
1	0.516	2	0	0.310811	1	0.334031	1	0
2	0.304	0	0	0.324324	8	0.636357	3	1
3	0.698	0	0	0.283784	1	0.000000	2	0
4	1.000	2	0	0.337838	2	0.500246	1	1

Over Sampling

In [28]:

```
X = df_scaled
sm  = SMOTE(random_state=42)
X_res, y_res = sm.fit_sample(X, y)
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test
_size= 0.2, random_state=7)
```

In [29]:

```
clf = XGBClassifier(max_depth = 12,random_state=7, n_estimators=100, e
val_metric = 'auc', min_child_weight = 3,
                    colsample_bytree = 0.75, subsample= 0.8)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1:", f1_score(y_test, y_pred))
print("Area under precision (AUC) Recall:", average_precision_score(y_
test, y_pred))
```

```
Accuracy: 0.8979912115505336
Precision: 0.9145631067961165
Recall: 0.8798256537982565
F1: 0.8968581402729293
Area under precision (AUC) Recall: 0.8652336100558795
```

In [30]:

```
# Confusion Matrix
confusion_matrix(y_test, y_pred)
```


Data

Data Sources

Deep Learning A-Z - ANN dataset

Churn_Modelling.csv 14 columns

Deep Learning A-Z - ANN dataset

Kirill Eremenko "Deep Learning A-Z™: Hands-On Artificial Neural Networks" course

Last Updated: 2 years ago (Version 1)

About this Dataset

Context

This is the dataset used in the section "ANN (Artificial Neural Networks)" of the Udemy course from Kirill Eremenko (Data Scientist & Forex Systems Expert) and Hadelin de Ponteves (Data Scientist), called **Deep Learning A-Z™: Hands-On Artificial Neural Networks**. The dataset is **very useful for beginners** of Machine Learning, and a simple playground where to compare several techniques/skills.

It can be freely downloaded here:
https://www.superdatascience.com/deep-learning/

The story: A bank is investigating a very high rate of customer leaving the bank. Here is a 10.000 records dataset to investigate and predict which of the customers are more likely to leave the bank soon.

The story of the story: I'd like to compare several techniques (better if not alone, and with the experience of several Kaggle users) to improve my basic knowledge on Machine Learning.

Content

I will write more later, but the columns names are very self-explaining.

Acknowledgements

Comments (2)

Sort by

All Comments | Hotness

Click here to comment...

DeepakChawla • Posted on Version 9 of 11 • a year ago • Options • Reply 1

@NasirIslamSujan, how you decide that these features are irrelivant RowNumber, CustomerId and Surname

Nasir Islam Sujan Kernel Author • Posted on Version 9 of 11 • a year ago • Options • Reply 0

@DeepakChawla

`RowNumber`, `CustomerId` and `Surname` are **Identical valued Feature**, means it contains unique values for each and every single observation.
Identical Feature does not make any **impact** or it does not **contribute** to our model instead it creates **bias/error** in our Model. So we simply avoid those feature.

Similar Kernels