In [1]:

```
# pip install xlrd pandas sklearn
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report
```

In [2]:

```
df = pd.read_excel("Kundenabwanderung.xlsx")
df["Umsatz"].fillna(df["Umsatz"].median(), inplace=True)
df["Land"] = df["Land"].factorize()[0]
df["Geschlecht"] = df["Geschlecht"].factorize()[0]
df.head()
```

Out[2]:

	RowNr	KundenID	Nachname	BonitaetsScore	Land	Geschlecht	Alter	Laufzeit	Umsa
0	1	15634602	Hargrave	619	0	0	42	2	119839.6
1	2	15647311	Hill	608	1	0	41	1	83807.8
2	3	15619304	Onio	502	0	0	42	8	159660.8
3	4	15701354	Boni	699	0	0	39	1	119839.6
4	5	15737888	Mitchell	850	1	0	43	2	125510.8

In [3]:

```
X = df.drop(["RowNr","KundenID", "Nachname", "Gekuendigt6M"], axis=1)
y = df["Gekuendigt6M"]

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.33, random_state=1
00)
```

Logistic Regression Classifier

In [4]:

```
LR = LogisticRegression(solver="sag", max_iter=10000, multi_class='multinomial')
LR.fit(X_train, y_train)
prediction = LR.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

```
accuracy: 79.27%
```

Linear Support Vector Classifier

In [5]:

```
LSVC = LinearSVC()
LSVC.fit(X_train, y_train)
prediction = LSVC.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

```
accuracy: 79.27%
```

```
C:\Users\amr.khalil\AppData\Local\Continuum\miniconda3\envs\gpu\lib\site-p
ackages\sklearn\svm\_base.py:977: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

Multinomial Naive Bayes Classifier

In [6]:

```
MNB = MultinomialNB()
MNB.fit(X_train, y_train)
prediction = MNB.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

```
accuracy: 50.39%
```

Bernoulli Naive Bayes Classifier

In [7]:

```
BNB = BernoulliNB()
BNB.fit(X_train, y_train)
prediction = BNB.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

```
accuracy: 79.27%
```

KNN Classifier

In [8]:

```
KNN = KNeighborsClassifier(n_neighbors = 70, weights = 'distance',algorithm = 'brute')
KNN.fit(X_train, y_train)
prediction = KNN.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

```
accuracy: 79.12%
```

Stochastic Gradient Descent

In [9]:

```
SGD = SGDClassifier(loss='squared_hinge',  alpha=0.0001, tol=0.1)
SGD.fit(X_train, y_train)
prediction = SGD.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

```
accuracy: 55.67%
```

Gradient Boost Classifier

In [10]:

```
GB = GradientBoostingClassifier()

GB.fit(X_train, y_train)
prediction = GB.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

```
accuracy: 86.61%
```

Random Forest Classifier

In [11]:

```
RF = RandomForestClassifier()
RF.fit(X_train, y_train)
prediction = RF.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

```
accuracy: 86.03%
```