Things Done:

1. Data has been cleaned and empty values are replaced with mean of tht column
2. Normalizing and Splitting of Data
3. Naive-Bayes
4. Knn
5. Random-Forest
6. SVM

Note:

1. We got rid of PCA and the plotting part as they were redundant

In [35]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn import svm
```

In [36]:
```
data = pd.read_excel('./data.xlsx')
data.head(25)
```

Out[36]:

	customer_id	name	age	gender	owns_car	owns_house	no_of_children	net_yearly_incor
0	CST_115179	ita Bose	46	F	N	Y	0.0	107934.
1	CST_121920	Alper Jonathan	29	M	N	Y	0.0	109862.
2	CST_109330	Umesh Desai	37	M	N	Y	0.0	230153.
3	CST_128288	Rie	39	F	N	Y	0.0	122325.
4	CST_151355	McCool	46	M	Y	Y	0.0	387286.
5	CST_123268	Sarah Marsh	46	F	Y	N	0.0	252765.
6	CST_127502	Mason	38	M	N	Y	1.0	262389.
7	CST_151722	Saba	46	F	Y	Y	1.0	241211.
8	CST_133768	Ashutosh	40	F	NaN	Y	0.0	210091.
9	CST_111670	David Milliken	39	F	Y	Y	2.0	207109.
10	CST_153773	Zaharia	32	F	N	Y	0.0	79102.
11	CST_142986	Sam	52	M	Y	N	1.0	158933.
12	CST_147654	Baker	39	F	N	Y	0.0	68421.
13	CST_165186	Vaughan	52	F	Y	N	0.0	125141.
14	CST_114892	Katharina Bart	43	M	Y	Y	0.0	368556.
15	CST_106781	Lesley Wroughton	37	F	N	N	0.0	746959.
16	CST_119686	Alister Bull	24	F	N	Y	0.0	145522.
17	CST_107195	Joan Biskupic	41	M	N	N	2.0	110975.
18	CST_162929	Smith	34	F	N	Y	0.0	198608.
19	CST_119824	Natalie Thomas	50	F	N	N	1.0	86956.
20	CST_144202	Huw	24	F	N	Y	0.0	123082.
21	CST_151332	James	36	F	N	Y	NaN	253922.
22	CST_105226	Tim Reid	42	M	Y	N	2.0	136743.
23	CST_109578	Poornima Gupta	23	F	N	Y	2.0	217697.
24	CST_119972	Jonathan Kaminsky	37	F	N	Y	0.0	95256.

In [37]:
```
data.shape
```

Out[37]:
```
(45528, 19)
```

In [38]:
```
data.describe()
```

Out[38]:

	age	no_of_children	net_yearly_income	no_of_days_employed	total_family_membe
count	45528.000000	44754.000000	4.552800e+04	45065.000000	45445.00000
mean	38.993411	0.420655	2.006556e+05	67609.289293	2.15808
std	9.543990	0.724097	6.690740e+05	139323.524434	0.91157
min	23.000000	0.000000	2.717061e+04	2.000000	1.00000
25%	31.000000	0.000000	1.263458e+05	936.000000	2.00000
50%	39.000000	0.000000	1.717149e+05	2224.000000	2.00000
75%	47.000000	1.000000	2.406038e+05	5817.000000	3.00000
max	55.000000	9.000000	1.407590e+08	365252.000000	10.00000

In [39]:
```
df = pd.DataFrame(data)
features=df.iloc[:,[2,7,8,12,13,15]]
labels=df.iloc[:,-1]
features
```

Out[39]:

	age	net_yearly_income	no_of_days_employed	yearly_debt_payments	credit_limit	credit_sc
0	46	107934.04	612.0	33070.28	18690.93	54
1	29	109862.62	2771.0	15329.53	37745.19	85
2	37	230153.17	204.0	48416.60	41598.36	65
3	39	122325.82	11941.0	22574.36	32627.76	75
4	46	387286.00	1459.0	38282.95	52950.64	92
...	...	...	...	...	...	
45523	55	96207.57	117.0	11229.54	29663.83	90
45524	31	383476.74	966.0	43369.91	139947.16	67
45525	27	260052.18	1420.0	22707.51	83961.83	72
45526	32	157363.04	2457.0	20150.10	25538.72	80
45527	38	316896.28	1210.0	34603.78	36630.76	68

45528 rows × 6 columns

In [40]:
```
# Replace zeroes with mean of the particular column
zero_not_accepted = ['age','net_yearly_income','no_of_days_employed','yearly_debt

for column in zero_not_accepted:
    features[column] = features[column].replace(np.NaN,0)
    mean = float(features[column].mean(skipna=True))
    features[column] = features[column].replace(0, mean)

features.describe()
# ignore the below warning
```

```
<ipython-input-40-7f053d45392d>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  features[column] = features[column].replace(np.NaN,0)
<ipython-input-40-7f053d45392d>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  features[column] = features[column].replace(0, mean)
```

Out[40]:

	age	net_yearly_income	no_of_days_employed	yearly_debt_payments	credit_limi
count	45528.000000	4.552800e+04	45528.000000	45528.000000	4.552800e+04
mean	38.993411	2.006556e+05	67602.297136	31796.826867	4.354842e+04
std	9.543990	6.690740e+05	138613.285749	17251.699947	1.487847e+05
min	23.000000	2.717061e+04	2.000000	2237.470000	4.003140e+03
25%	31.000000	1.263458e+05	946.750000	19240.262500	2.397381e+04
50%	39.000000	1.717149e+05	2261.000000	29122.265000	3.568804e+04
75%	47.000000	2.406038e+05	6206.000000	40535.472500	5.343576e+04
max	55.000000	1.407590e+08	365252.000000	328112.860000	3.112997e+07

In [41]:
```
labels
```

Out[41]:
```
0        1
1        0
2        0
3        0
4        0
        ..
45523    0
45524    0
45525    0
45526    0
45527    0
Name: credit_card_default, Length: 45528, dtype: int64
```

In [42]:
```
# splitting the data into train and test
X_train, X_test, Y_train, Y_test = train_test_split(features, labels, test_size=0
print(len(X_train),len(X_test))
```

```
34146 11382
```

In [43]:
```
# normalizing all the values so that every value can have equal contribution fact
sc_x = StandardScaler()
X_train = sc_x.fit_transform(X_train)
X_test = sc_x.transform(X_test)
```

In [44]:
```
# training a Naive Bayes classifier
gnb = GaussianNB().fit(X_train, Y_train)
gnb_predictions = gnb.predict(X_test)
accuracy = gnb.score(X_test, Y_test)
print(accuracy)
cm = confusion_matrix(Y_test, gnb_predictions)
print(cm)
```

```
0.9102091020910209
[[10341    87]
 [  935    19]]
```

In [45]:
```
# training a KNN model
Classifier = KNeighborsClassifier(n_neighbors=128)
Classifier.fit(X_train, Y_train)
ypredicted = Classifier.predict(X_test)
accuracy=Classifier.score(X_test,Y_test)
print(accuracy)
cm = confusion_matrix(Y_test, ypredicted)
print(cm)
```

```
0.9649446494464945
[[10428     0]
 [  399   555]]
```

In [46]:
```
# Training a Random Forest Classifier
RF = RandomForestClassifier(n_estimators = 100)
RF.fit(X_train, Y_train)

y_pred = RF.predict(X_test)
accuracy = RF.score(X_test,Y_test)
print(accuracy)
cm = confusion_matrix(Y_test, y_pred)
print(cm)
```

```
0.979353364962221
[[10425     3]
 [  232   722]]
```

In [47]:
```
from sklearn.model_selection import train_test_split
# Building a Support Vector Machine on train data
svc_model = svm.SVC(C= .1, kernel='linear', gamma= 1)
svc_model.fit(X_train, Y_train)
y_prediction = svc_model.predict(X_test)
accuracy = svc_model.score(X_test,Y_test)
print(accuracy)
cm = confusion_matrix(Y_test, y_prediction)
print(cm)
```

```
0.9731154454401687
[[10428     0]
 [  306   648]]
```

In [ ]:

In [ ]: