```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set_theme(color_codes=True)
```

You can get the source code here : https://www.kaggle.com/datasets/kmldas/loan-default-prediction?resource=download

```
df = pd.read_csv('Default_Fin.csv')
df.head()
```

	Index	Employed	Bank Balance	Annual Salary	Defaulted?
0	1	1	8754.36	532339.56	0
1	2	0	9806.16	145273.56	0
2	3	1	12882.60	381205.68	0
3	4	1	6351.00	428453.88	0
4	5	1	9427.92	461562.00	0

Exploratory Data Analysis

```
df.isnull().sum()
```

```
Index            0
Employed         0
Bank Balance     0
Annual Salary    0
Defaulted?       0
dtype: int64
```

```
df2 = df.drop('Index', axis=1)
df2.head()
```

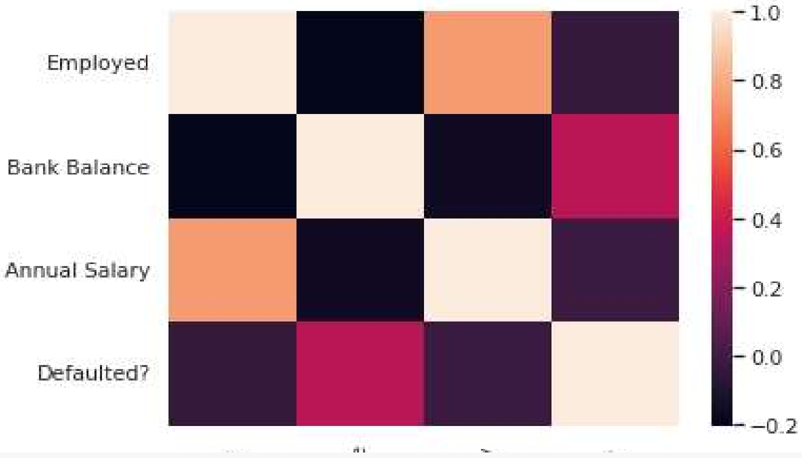
	Employed	Bank Balance	Annual Salary	Defaulted?
0	1	8754.36	532339.56	0
1	0	9806.16	145273.56	0
2	1	12882.60	381205.68	0
3	1	6351.00	428453.88	0
4	1	9427.92	461562.00	0

```
df2.dtypes
```

```
Employed           int64
Bank Balance     float64
Annual Salary    float64
Defaulted?         int64
dtype: object
```

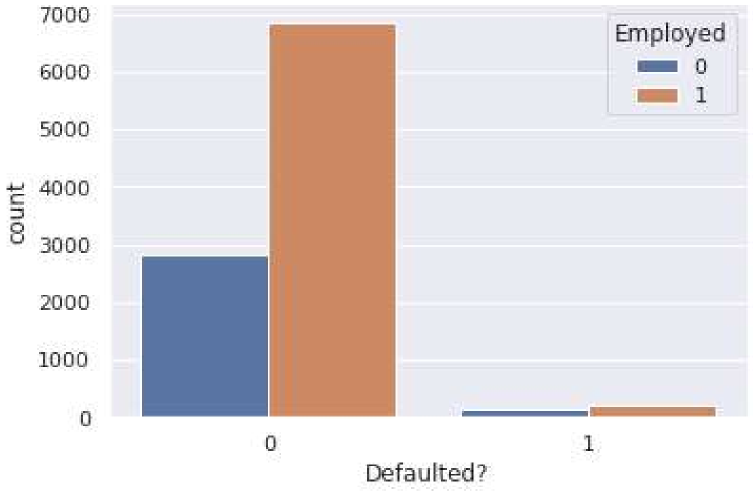
```
sns.heatmap(df2.corr(), fmt='.2g')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8b923b4a10>
```



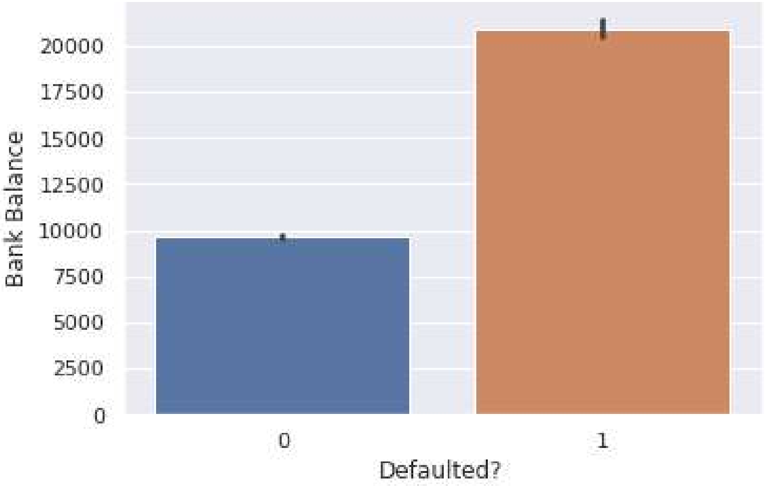
```
sns.countplot(data=df2, x="Defaulted?", hue="Employed")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8b8facaa50>
```



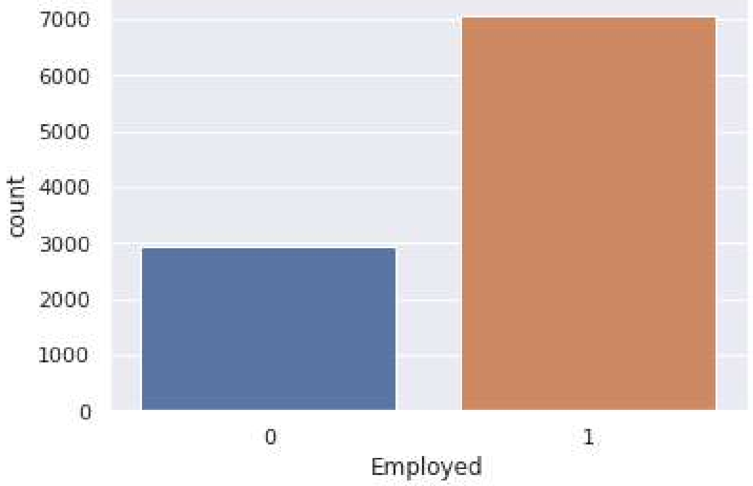
```
sns.barplot(data=df2, x="Defaulted?", y="Bank Balance")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8b8f595990>
```



```
sns.countplot(data=df2, x="Employed")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8b8f52c350>
```



Build Machine Learning Model

```
X = df.drop('Defaulted?', axis=1)
y = df['Defaulted?']
```

```
#test size 20% and train size 80%
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2,random_state=7)
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

```
y_pred = rfc.predict(X_test)
print("Accuracy Score :", accuracy_score(y_test, y_pred)*100, "%")
```

```
Accuracy Score : 97.05 %
```

XGBoost

```
from xgboost import XGBClassifier

xgb_model = XGBClassifier()
xgb_model.fit(X_train, y_train)
```

```
XGBClassifier()
```

```
y_pred = rfc.predict(X_test)
print("Accuracy Score :", accuracy_score(y_test, y_pred)*100, "%")
```

```
Accuracy Score : 97.05 %
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

```
y_pred = dtree.predict(X_test)
print("Accuracy Score :", accuracy_score(y_test, y_pred)*100, "%")
```

```
Accuracy Score : 95.45 %
```

Support Vector Machine

```python
from sklearn import svm

svm = svm.SVC()
svm.fit(X_train, y_train)
```

```
SVC()
```

```python
y_pred = svm.predict(X_test)
print("Accuracy Score :", accuracy_score(y_test, y_pred)*100, "%")
```

```
Accuracy Score : 96.65 %
```

Logistic Regression

```python
from sklearn.linear_model import LogisticRegression

lr = RandomForestClassifier()
lr.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

```python
y_pred = lr.predict(X_test)
print("Accuracy Score :", accuracy_score(y_test, y_pred)*100, "%")
```

```
Accuracy Score : 97.1 %
```

Naive Bayes

```python
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()
nb.fit(X_train, y_train)
```

```
GaussianNB()
```

```python
y_pred = nb.predict(X_test)
print("Accuracy Score :", accuracy_score(y_test, y_pred)*100, "%")
```

```
Accuracy Score : 97.25 %
```

K Nearest Neighbor

```python
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier()
```

```python
y_pred = knn.predict(X_test)
print("Accuracy Score :", accuracy_score(y_test, y_pred)*100, "%")
```

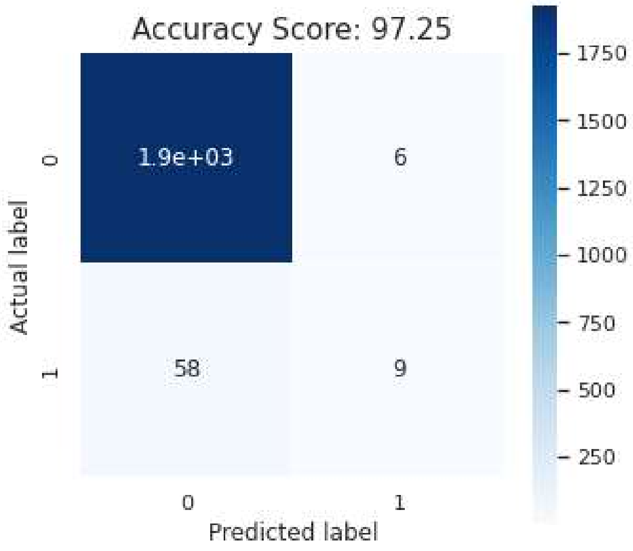
```
Accuracy Score : 96.8 %
```

Visualize Naive Bayes Algorithm

```
#importing classification report and confussion matrix from sklearn
from sklearn.metrics import classification_report, confusion_matrix
```

```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True,  cmap = 'Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0}'.format(nb.score(X_test, y_test)*100)
plt.title(all_sample_title, size = 15)
```

```
Text(0.5, 1.0, 'Accuracy Score: 97.25')
```



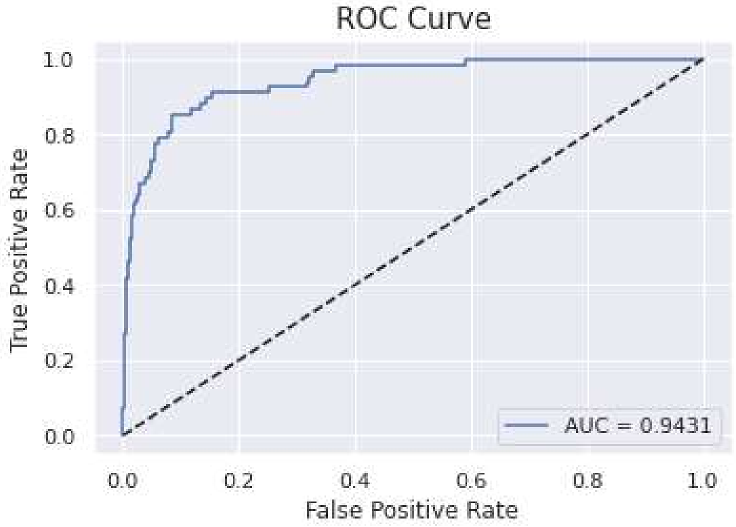
```
from sklearn.metrics import roc_curve, roc_auc_score
y_pred_proba = nb.predict_proba(X_test)[:][:,1]

df_actual_predicted = pd.concat([pd.DataFrame(np.array(y_test), columns=['y_actual']), pd.DataF
df_actual_predicted.index = y_test.index

fpr, tpr, tr = roc_curve(df_actual_predicted['y_actual'], df_actual_predicted['y_pred_proba'])
auc = roc_auc_score(df_actual_predicted['y_actual'], df_actual_predicted['y_pred_proba'])

plt.plot(fpr, tpr, label='AUC = %0.4f' %auc)
plt.plot(fpr, fpr, linestyle = '--', color='k')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve', size = 15)
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f8b82561cd0>
```



```
#Printing 10 first probabilities
y_pred_prob = nb.predict_proba(X_test)[0:10]

y_pred_prob
```

```
array([[9.39398000e-01, 6.06019997e-02],
       [9.99923288e-01, 7.67117235e-05],
       [8.86039124e-01, 1.13960876e-01],
       [9.16054768e-01, 8.39452322e-02],
       [9.77934396e-01, 2.20656044e-02],
       [9.99939652e-01, 6.03483322e-05],
       [9.99150496e-01, 8.49503566e-04],
       [9.95534289e-01, 4.46571067e-03],
       [9.99761280e-01, 2.38720148e-04],
       [9.95746555e-01, 4.25344519e-03]])
```

```
# store the probabilities in dataframe (first 10 data)

y_pred_prob_df = pd.DataFrame(data=y_pred_prob, columns=['Probability of Defaulted 0', 'Probabi

y_pred_prob_df
```

	Probability of Defaulted 0	Probability of Defaulted 1
0	0.939398	0.060602
1	0.999923	0.000077
2	0.886039	0.113961
3	0.916055	0.083945
4	0.977934	0.022066
5	0.999940	0.000060
6	0.999150	0.000850
7	0.995534	0.004466
8	0.999761	0.000239
9	0.995747	0.004253

Predicting data using Gaussian Naive Bayes

```
print(nb.predict([[1, 80000, 100000, 0]]))
```

```
[1]
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have val
  "X does not have valid feature names, but"
```

✓ 0s completed at 1:49 AM ● ×