Importing the libraries

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

Data collection and processing

```
# Loading the dataset into pandas dataframe
loan_dataset = pd.read_csv('/content/sample_data/Loan.csv.csv')
```

```
type(loan_dataset)
```

```
pandas.core.frame.DataFrame
```

```
# printing the first 5 rows of dataframe
loan_dataset.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amo
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	

```
# number of rows and columns in the dataset
loan_dataset.shape
```

```
(614, 13)
```

```
# number of values missing
loan_dataset.isnull().sum()
```

```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

```
# dropping the missing values
loan_dataset = loan_dataset.dropna()
```

```
# again,checking number of values missing
loan_dataset.isnull().sum()
```

```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
```

```
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

```
# label encoding
loan_dataset.replace({"Loan_Status":{'N':0,'Y':1}},inplace=True)
```

```
#printing first 5 values of dataset
loan_dataset.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amo
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	

```
# Dependent column values
loan_dataset['Dependents'].value_counts()
```

```
0     274
2      85
1      80
3+     41
Name: Dependents, dtype: int64
```

```
# replacing the value 3+ to 4
loan_dataset = loan_dataset.replace(to_replace='3+',value=4)
```

```
# dependent values
loan_dataset['Dependents'].value_counts()
```

```
0    274
2     85
1     80
4     41
Name: Dependents, dtype: int64
```

```
# convert categorical columns to numerical values
loan_dataset.replace({'Married':{'No':0,'Yes':1},'Gender':{'Male':1,'Female':0},'Self_Employed':{'No':0,'Yes':1},
                      'Property_Area':{'Rural':0,'Semiurban':1,'Urban':2},'Education':{'Graduate':1,'Not Graduate':0}},inplace=True)
```

```
loan_dataset.head()
```

rried	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
1	1	1	0	4583	1508.0	128.0	360.0	1.0
1	0	1	1	3000	0.0	66.0	360.0	1.0
1	0	0	0	2583	2358.0	120.0	360.0	1.0
0	0	1	0	6000	0.0	141.0	360.0	1.0
1	2	1	1	5417	4196.0	267.0	360.0	1.0

```
# seperating the data and label
X = loan_dataset.drop(columns=['Loan_ID','Loan_Status'],axis=1)
Y = loan_dataset['Loan_Status']
```

```
print(X)
print(Y)
```

```
     Gender  Married Dependents  Education  Self_Employed  ApplicantIncome  \
1         1        1          1          1              0             4583
2         1        1          0          1              1             3000
3         1        1          0          0              0             2583
4         1        0          0          1              0             6000
5         1        1          2          1              1             5417
..      ...      ...        ...        ...            ...              ...
609       0        0          0          1              0             2900
610       1        1          4          1              0             4106
611       1        1          1          1              0             8072
612       1        1          2          1              0             7583
613       0        0          0          1              1             4583

     CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
1               1508.0       128.0             360.0             1.0
2                  0.0        66.0             360.0             1.0
3               2358.0       120.0             360.0             1.0
4                  0.0       141.0             360.0             1.0
5               4196.0       267.0             360.0             1.0
..                 ...         ...               ...             ...
609                0.0        71.0             360.0             1.0
610                0.0        40.0             180.0             1.0
611              240.0       253.0             360.0             1.0
612                0.0       187.0             360.0             1.0
613                0.0       133.0             360.0             0.0

     Property_Area
1                0
2                2
3                2
4                2
5                2
..             ...
609              0
610              0
611              2
612              2
613              1

[480 rows x 11 columns]
1      0
2      1
3      1
4      1
5      1
      ..
609    1
610    1
611    1
612    1
613    0
Name: Loan_Status, Length: 480, dtype: int64
```

Data Standardization

```
1 scaler = StandardScaler()
2 scaler.fit(X)
```

▾ StandardScaler

StandardScaler()

```
1 standardized_data=scaler.transform(X)
2 print(standardized_data)
```

```
[[ 0.46719815  0.73716237  0.11235219 ...  0.27554157  0.41319694
  -1.31886834]
 [ 0.46719815  0.73716237 -0.70475462 ...  0.27554157  0.41319694
   1.25977445]
 [ 0.46719815  0.73716237 -0.70475462 ...  0.27554157  0.41319694
   1.25977445]
 ...
 [ 0.46719815  0.73716237  0.11235219 ...  0.27554157  0.41319694
   1.25977445]
 [ 0.46719815  0.73716237  0.92945899 ...  0.27554157  0.41319694
   1.25977445]
 [-2.14041943 -1.35655324 -0.70475462 ...  0.27554157 -2.42015348
  -0.02954695]]
```

```
X=standardized_data
Y=loan_dataset['Loan_Status']
print(X)
print(Y)
```

```
[[ 0.46719815  0.73716237  0.11235219 ...  0.27554157  0.41319694
  -1.31886834]
 [ 0.46719815  0.73716237 -0.70475462 ...  0.27554157  0.41319694
   1.25977445]
 [ 0.46719815  0.73716237 -0.70475462 ...  0.27554157  0.41319694
   1.25977445]
 ...
 [ 0.46719815  0.73716237  0.11235219 ...  0.27554157  0.41319694
   1.25977445]
 [ 0.46719815  0.73716237  0.92945899 ...  0.27554157  0.41319694
   1.25977445]
 [-2.14041943 -1.35655324 -0.70475462 ...  0.27554157 -2.42015348
  -0.02954695]]
1      0
2      1
3      1
4      1
5      1
      ..
609    1
610    1
611    1
612    1
613    0
Name: Loan_Status, Length: 480, dtype: int64
```

Train and Test split

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,stratify=Y,random_state=2)
```

```
print(X.shape,X_train.shape,X_test.shape)
```

```
(480, 11) (432, 11) (48, 11)
```

Traning the model

- SUPPORT VECTOR MACHINE

```
classifier=svm.SVC(kernel='linear')
```

```
# Training the support vector machine
classifier.fit(X_train,Y_train)
```

```
SVC(kernel='linear')
```

```
# accuracy score on training data
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction,Y_train)
print('Acdurracy on training data : ',training_data_accuracy )
```

```
Acdurracy on training data :  0.7986111111111112
```

```
# accuracy score on test data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction,Y_test)
print('Accurracy on test data : ',test_data_accuracy )
```

```
Accurracy on test data :  0.8333333333333334
```

Making a predictive system

```
input_data=(1,1,0,1,1,3000,0.0,66.0,360.0,1.0,2)

# changing the input data to numpy array
input_data_as_numpy_array=np.asarray(input_data)

```

```
# reshape the data as we are predicting the label for only one instance
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)

# Standardization the input
std_data=scaler.transform(input_data_reshaped)
print(std_data)
prediction=classifier.predict(std_data)
print(prediction)
if(prediction[0]==0):
  print('Loan status not approved')
else:
  print('Loan status approved')
```

```
[[ 0.46719815  0.73716237 -0.70475462  0.50325312  2.50454133 -0.4175358
  -0.604633   -0.97900085  0.27554157  0.41319694  1.25977445]]
[1]
Loan status approved
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was fi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but SVC was fitted with f
  warnings.warn(
```

```
input_data=(1,0,1,1,0,5000,0.3,53.0,360.0,0,1)

# changing the input data to numpy array
input_data_as_numpy_array=np.asarray(input_data)

# reshape the data as we are predicting the label for only one instance
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)

# Standardization the input
std_data=scaler.transform(input_data_reshaped)
print(std_data)
prediction=classifier.predict(std_data)
print(prediction)
if(prediction[0]==0):
  print('Loan status not approved')
else:
  print('Loan status approved')
```

```
[[ 0.46719815 -1.35655324  0.11235219  0.50325312 -0.3992747  -0.06432517
  -0.60451828 -1.14064362  0.27554157 -2.42015348 -0.02954695]]
[0]
Loan status not approved
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was fi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but SVC was fitted with f
  warnings.warn(
```