```
import numpy as np
 import pandas as pd
 import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score

# loading the dataset to pandas DataFrame
loan_dataset = pd.read_csv('/content/dataset.csv')
```

```
loan_dataset.head()

# number of rows and columns
loan_dataset.shape

# statistical measures
loan_dataset.describe()

# number of missing values in each column
loan_dataset.isnull().sum()

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

# number of missing values in each column
loan_dataset.isnull().sum()

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

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

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

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

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

# education & Loan Status
```

```
sns.countplot(x='Education',hue='Loan_Status',data=loan_dataset)

# marital status & Loan Status
sns.countplot(x='Married',hue='Loan_Status',data=loan_dataset)

# convert categorical columns to numerical values
loan_dataset.replace({'Married':{'No':0,'Yes':1},'Gender':{'Male':1,'Fe
male':0},'Self_Employed':{'No':0,'Yes':1},

'Property_Area':{'Rural':0,'Semiurban':1,'Urban':2},'Education':{'Grad
uate':1,'Not Graduate':0}},inplace=True)

loan_dataset.head()

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

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

#training the support Vector Macine model
classifier.fit(X_train,Y_train)

# accuracy score on training data
X_train_prediction = classifier.predict(X_train)
training_data_accuray = accuracy_score(X_train_prediction,Y_train)

print('Accuracy on training data : ', training_data_accuray)

# accuracy score on training data
X_test_prediction = classifier.predict(X_test)
```

```
test_data_accuray = accuracy_score(X_test_prediction,Y_test)

print('Accuracy on test data : ', test_data_accuray)

input_data =(1,1,1,1,0,4583,1508.0,128.0,360.0,1.0,0)

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

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

# standardize the input data
std_data = scaler.transform(input_data_reshaped)
print(std_data)

prediction = classifier.predict(std_data)
print(prediction)

if (prediction[0] == 0):
  print('The person is not eligible to get loan')
else:
  print('The person is eligible to get loan')



loan_dataset.head()

# number of rows and columns
loan_dataset.shape

# statistical measures
loan_dataset.describe()

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

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

# number of missing values in each column
loan_dataset.isnull().sum()

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

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

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

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

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

# education & Loan Status
```

```
sns.countplot(x='Education',hue='Loan_Status',data=loan_dataset)

# marital status & Loan Status
sns.countplot(x='Married',hue='Loan_Status',data=loan_dataset)

# convert categorical columns to numerical values
loan_dataset.replace({'Married':{'No':0,'Yes':1},'Gender':{'Male':1,'Fe
male':0},'Self_Employed':{'No':0,'Yes':1},

'Property_Area':{'Rural':0,'Semiurban':1,'Urban':2},'Education':{'Grad
uate':1,'Not Graduate':0}},inplace=True)

loan_dataset.head()

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

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

#training the support Vector Macine model
classifier.fit(X_train,Y_train)

# accuracy score on training data
X_train_prediction = classifier.predict(X_train)
training_data_accuray = accuracy_score(X_train_prediction,Y_train)

print('Accuracy on training data : ', training_data_accuray)

# accuracy score on training data
X_test_prediction = classifier.predict(X_test)
```

```
test_data_accuray =

accuracy_score(X_test_prediction,Y_test)print('Accuracy

on test data : ', test_data_accuray)

input_data =(1,1,1,1,0,4583,1508.0,128.0,360.0,1.0,0)

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

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

# standardize the input data
std_data = scaler.transform(input_data_reshaped)
print(std_data)

prediction =
classifier.predict(std_data)
print(prediction)

if (prediction[0] == 0):
  print('The person is not eligible to get
loan')else:
  print('The person is eligible to get loan')
```