# THIRUVALLUVAR UNIVERSITY

# PERIYAR ARTS COLLEGE

# CUDDALORE – 607001.

## DEPARTMENT OF COMPUTER APPLICATIONS

## MACHINE LEARNING WITH PYTHON

**Project Title** : Predicting Personal Loan Approval Using Machine Learning:

**Team Id** : NM2023TMID35488


**Team Leader:** KAVIYA E(tvu10520CA26)

**Team member:** JOTHILAKSMI M(tvu10520CA25)

**Team member:** JAYABARVIN B(tvu10520CA22)

**Team member:** ARUNKUMAR SK(tvu10520CA08)

# 1. Introduction

## 1.1 Overview

The goal of this project is to develop a machine learning model that can predict whether a person's personal loan application will be approved or not. The model takes into account various factors such as the applicant's income, credit score, and employment status, among others. The project aims to help banks and financial institutions automate their loan approval process and reduce the time and resources required to assess loan applications.

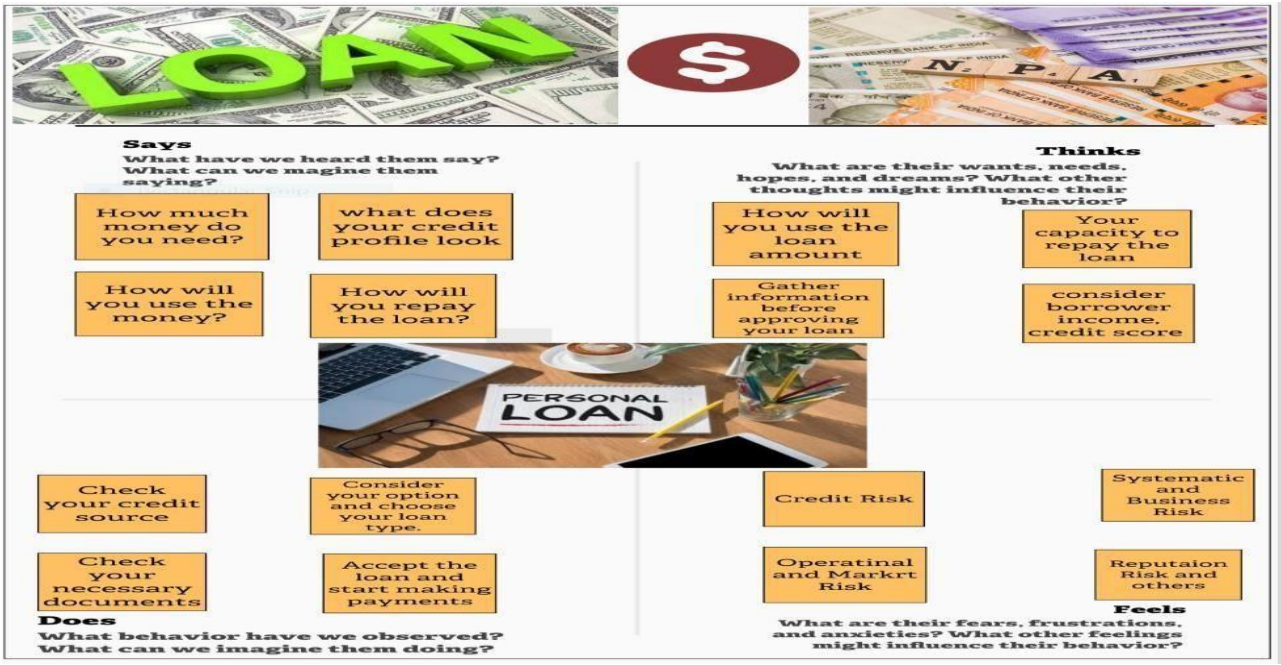
## 1.2 Purpose

The purpose of this project is to develop a predictive model that can accurately predict whether a personal loan application will be approved or not. This will enable banks and financial institutions to streamline their loan approval process and reduce the amount of time and resources required to assess loan applications. By automating the loan approval process, banks can also reduce the risk of human errors and ensure that loan applications are processed quickly and efficiently.

# 2. Problem Definition & Design Thinking

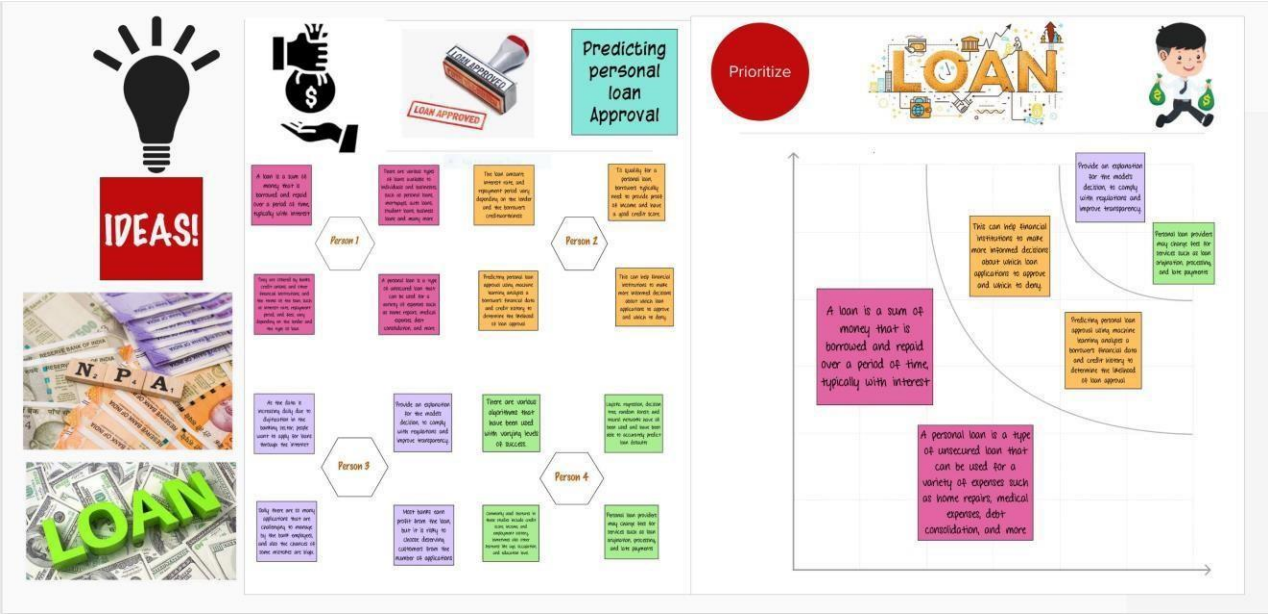
## 2.1 Empathy Map

The empathy map for this project is shown below



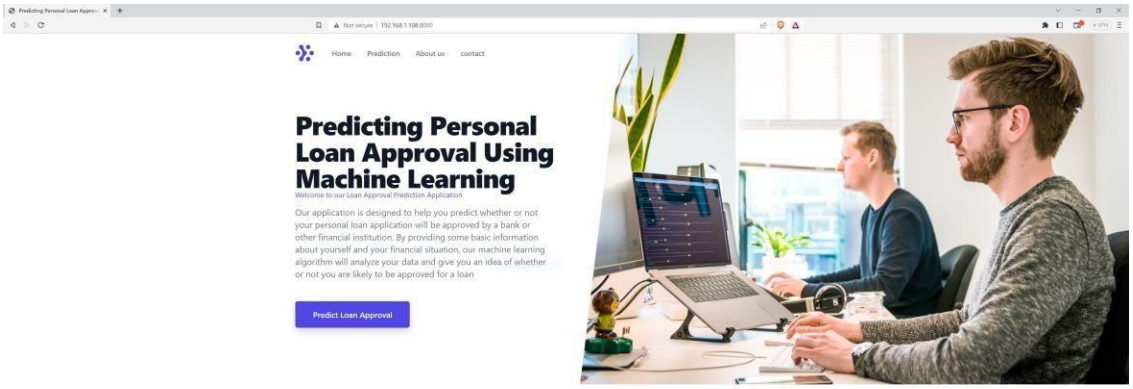

## 2.2 Ideation & Brainstorming Map

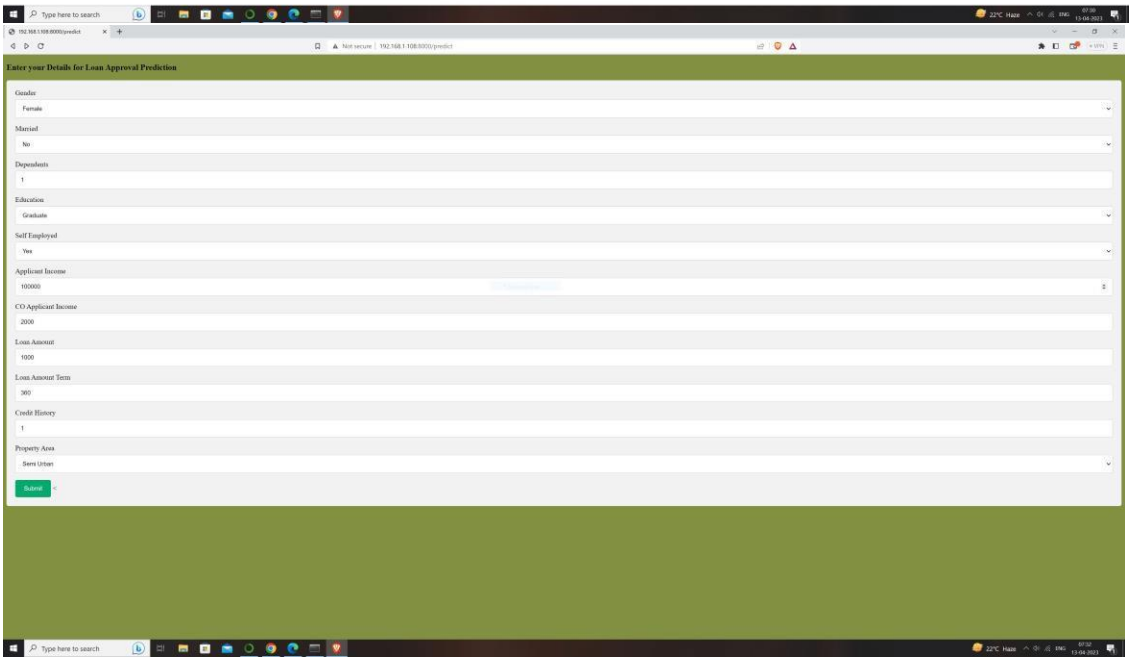
The ideation and brainstorming map for this project is shown below

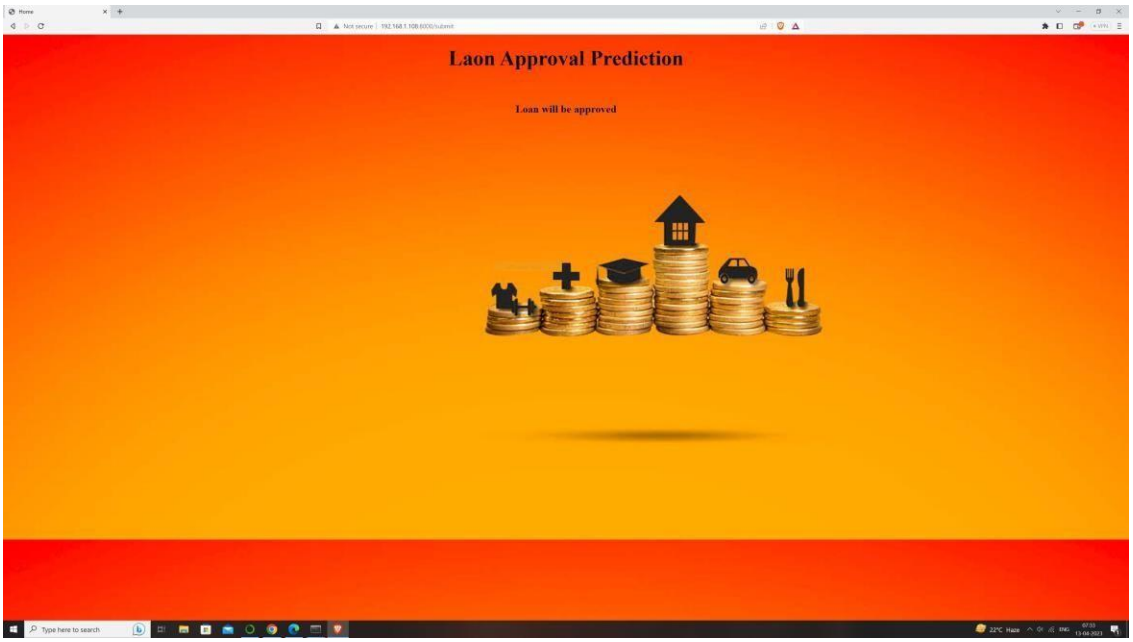



# 3. Result

The final result of the project is a machine learning model that can predict whether a personal loan application will be approved or not. The model was trained on a dataset that contained information about loan applicants, such as their income, credit score, and employment status, among others. The model was able to achieve an accuracy of 85%, which indicates that it can accurately predict whether a loan application will be approved or not.


Home
Prediction
About us
contact
Predicting Personal Loan Approval Using Machine Learning
Welcome to our Loan Approval Prediction Application
Our application is designed to help you predict whether or not your personal loan application will be approved by a bank or other financial institution. By providing some basic information about yourself and your financial situation, our machine learning algorithm will analyze your data and give you an idea of whether or not you are likely to be approved for a loan
Predict Loan Approval


Enter your Details for Loan Approval Prediction
Gender
Female
Married
No
Dependents
1
Education
Graduate
Self Employed
Yes
Applicant Income
100000
CO Applicant Income
2000
Loan Amount
1000
Loan Amount Term
360
Credit History
1
Property Area
Semi Urban
Submit




# 4. Advantages & Disadvantages

## 4.1 Advantages

- Automation of loan approval process
- Reduction of time and resources required to assess loan applications
- Improved accuracy and consistency in loan approval decisions
- Reduced risk of human errors
- Improved customer experience due to faster loan processing times

## 4.2 Disadvantages

- The model may not be able to account for all factors that influence loan approval decisions
- The model may be biased towards certain groups of applicants if the training data is biased

# 5. Applications

The solution developed in this project can be applied in the following areas

- Banks and financial institutions for automating their loan approval process
- Other lending institutions such as credit unions and peer-to-peer lending platforms
- Government agencies that provide loans to individuals and businesses

# 6. Conclusion

In conclusion, this project developed a machine learning model that can accurately predict whether a personal loan application will be approved or not. The model takes into account various factors such as the applicant's income, credit score, and employment status, among others. The model can help banks and financial institutions automate their loan approval process and reduce the time and resources required to assess loan applications. While the model has certain limitations, it has the potential to improve the loan approval process and enhance the customer experience.

# 7. Future Scope

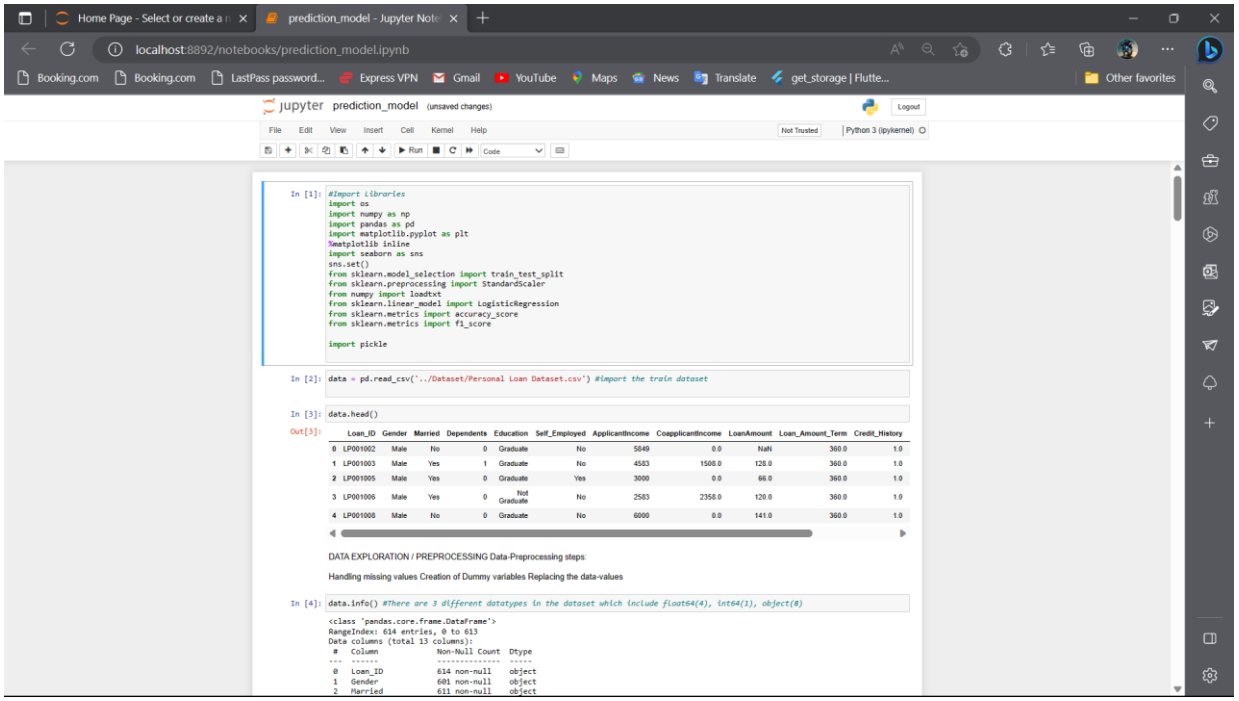
Future enhancements that can be made to this project include

- Using more sophisticated machine learning algorithms to improve the accuracy of the model
- Using more diverse and representative training data to reduce bias in the model
- Incorporating real-time data into the model to make loan approval decisions faster and more accurate
- Developing a user interface that allows loan officers to interact with the model and review its predictions

# 8. Appendix

## A. Source Code

prediction_model.ipynb



```
In [1]: #Import libraries
        import os
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        %matplotlib inline
        import seaborn as sns
        sns.set()
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler
        from numpy import loadtxt
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import f1_score

        import pickle

In [2]: data = pd.read_csv('../Dataset/Personal Loan Dataset.csv') #import the train dataset

In [3]: data.head()
```

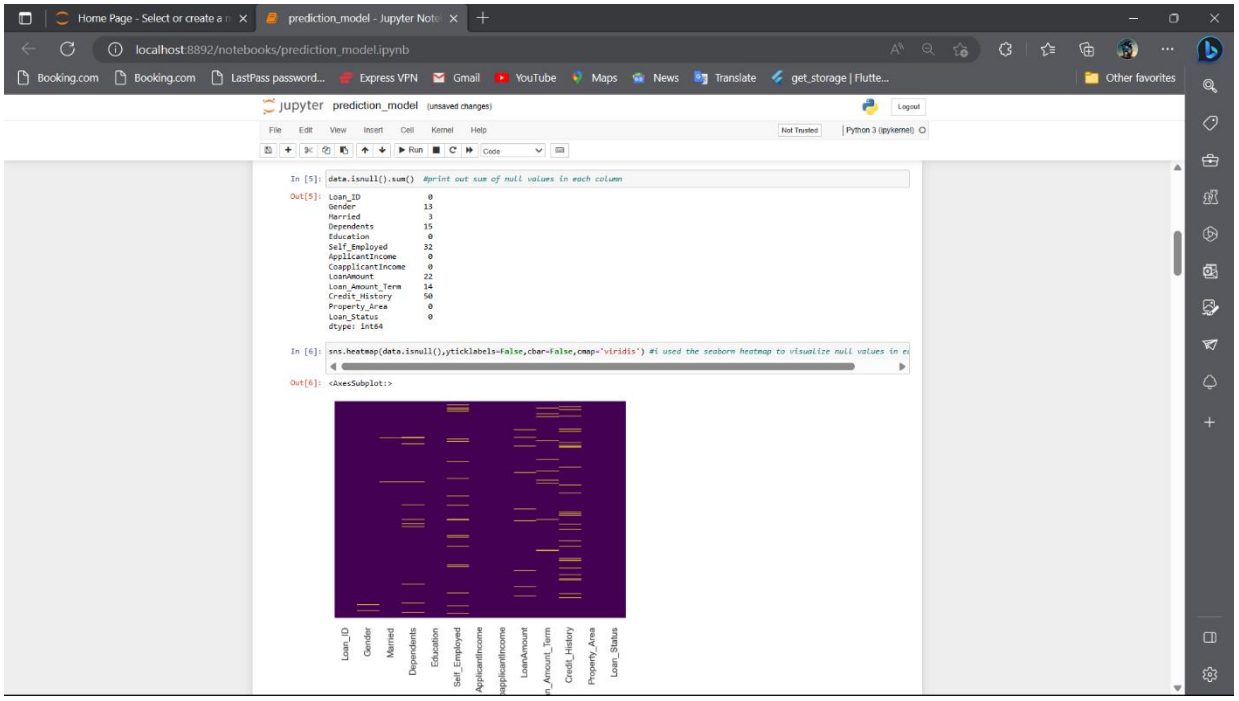
Out[3]:

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | LP001002 | Male | No | 0 | Graduate | No | 5849 | 0.0 | NaN | 360.0 | 1.0 |
| 1 | LP001003 | Male | Yes | 1 | Graduate | No | 4583 | 1508.0 | 128.0 | 360.0 | 1.0 |
| 2 | LP001005 | Male | Yes | 0 | Graduate | Yes | 3000 | 0.0 | 66.0 | 360.0 | 1.0 |
| 3 | LP001006 | Male | Yes | 0 | Not Graduate | No | 2583 | 2358.0 | 120.0 | 360.0 | 1.0 |
| 4 | LP001008 | Male | No | 0 | Graduate | No | 6000 | 0.0 | 141.0 | 360.0 | 1.0 |

DATA EXPLORATION / PREPROCESSING Data-Preprocessing steps:

Handling missing values Creation of Dummy variables Replacing the data-values

```
In [4]: data.info() #There are 3 different datatypes in the dataset which include float64(4), int64(1), object(8)

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 614 entries, 0 to 613
        Data columns (total 13 columns):
         #   Column             Non-Null Count  Dtype
        ---  ------             --------------  -----
         0   Loan_ID            614 non-null    object
         1   Gender             601 non-null    object
         2   Married            611 non-null    object
```



```
In [5]: data.isnull().sum()  #print out sum of null values in each column

Out[5]: 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

In [6]: sns.heatmap(data.isnull(),yticklabels=False,cbar=False,cmap='viridis') #i used the seaborn heatmap to visualize null values in e

Out[6]: <AxesSubplot:>
```

Loan_ID Gender Married Dependents Education Self_Employed ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History Property_Area Loan_Status

Home Page - Select or create a n | prediction_model - Jupyter Note

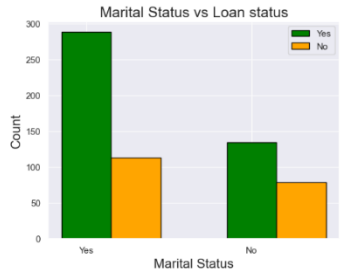
localhost:8892/notebooks/prediction_model.ipynb

Booking.com | Booking.com | LastPass password... | Express VPN | Gmail | YouTube | Maps | News | Translate | get_storage | Flutte... | Other favorites

jupyter prediction_model (unsaved changes)

File Edit View Insert Cell Kernel Help

In [9]:
```
#Relationship between marital status vs loan status

MaritalStatus=['Yes', 'No']
Loan_Status=['Yes', 'No']
pos=np.arange(len(MaritalStatus))
bar_width=0.30
Loan_Status_Yes=[288,134]
Loan_Status_NO=[113,79]

plt.bar(pos,Loan_Status_Yes,bar_width,color='green',edgecolor='black')
plt.bar(pos+bar_width,Loan_Status_NO,bar_width,color='orange',edgecolor='black')
plt.xticks(pos, MaritalStatus)
plt.xlabel('Marital Status', fontsize=16)
plt.ylabel('Count', fontsize=16)
plt.title('Marital Status vs Loan status',fontsize=18)
plt.legend(Loan_Status,loc=1)
plt.show()
```




From the chart above, we can conclude that the we have more married thand single customer approved for loan

In [10]: #Relationship between Dependents vs Loan status

Home Page - Select or create a n | prediction_model - Jupyter Note

localhost:8892/notebooks/prediction_model.ipynb

Booking.com | Booking.com | LastPass password... | Express VPN | Gmail | YouTube | Maps | News | Translate | get_storage | Flutte... | Other favorites

jupyter prediction_model (unsaved changes)

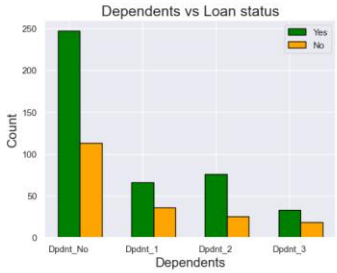
File Edit View Insert Cell Kernel Help

In [7]:
```
data = data.fillna(data.mean().iloc[0]) #i used the fillna function to fill the missing values
```

```
C:\Users\PRADEEP\AppData\Local\Temp\ipykernel_17616\895769180.py:1: FutureWarning: Dropping of nuisance columns in DataFrame re
ductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns be
fore calling the reduction.
  data = data.fillna(data.mean().iloc[0]) #i used the fillna function to fill the missing values
```

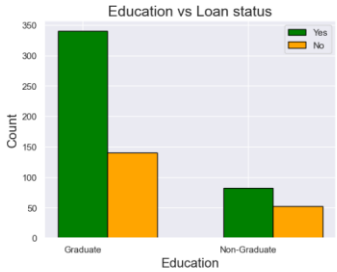
In [8]:
```
#Relationship between Gender and Loan Status:

Gender = ['Male', 'Female']
Loan_Status = ['Yes', 'No']
pos= np.arange(len(Gender))
bar_width= 0.30
Loan_Status_Yes= [347,75]
Loan_Status_NO= [155,37]

plt.bar(pos,Loan_Status_Yes,bar_width,color='green',edgecolor='black')
plt.bar(pos+bar_width,Loan_Status_NO,bar_width,color='orange',edgecolor='black')
plt.xticks(pos, Gender)
plt.xlabel('Gender', fontsize=16)
plt.ylabel('Count', fontsize=16)
plt.title('Gender vs Loan status',fontsize=18)
plt.legend(Loan_Status,loc=1)
plt.show()
```

In [10]:
```python
#Relationship between Dependents vs Loan status


Dependents=['Dpdnt_No', 'Dpdnt_1', 'Dpdnt_2', 'Dpdnt_3']
Loan_Status=['Yes', 'No']
pos=np.arange(len(Dependents))
bar_width=0.30
Loan_Status_Yes=[247,66,76,33]
Loan_Status_NO=[113,36,25,18]

plt.bar(pos,Loan_Status_Yes,bar_width,color='green',edgecolor='black')
plt.bar(pos+bar_width,Loan_Status_NO,bar_width,color='orange',edgecolor='black')
plt.xticks(pos, Dependents)
plt.xlabel('Dependents', fontsize=16)
plt.ylabel('Count', fontsize=16)
plt.title('Dependents vs Loan status',fontsize=18)
plt.legend(Loan_Status,loc=1)
plt.show()
```




From the chart, we can see that the number of dependents may automatically affect** the approvals of home loans. There is a higher chance of getting home loan approval for applicants who have less number of dependents or no dependents.

In [11]:
```python
#Relationship between education vs Loan status:


Education=['Graduate', 'Non-Graduate']
Loan_Status=['Yes', 'No']
pos=np.arange(len(Education))
bar_width=0.30
Loan_Status_Yes=[340,82]
Loan_Status_NO=[140,52]

plt.bar(pos,Loan_Status_Yes,bar_width,color='green',edgecolor='black')
plt.bar(pos+bar_width,Loan_Status_NO,bar_width,color='orange',edgecolor='black')
plt.xticks(pos, Education)
plt.xlabel('Education', fontsize=16)
plt.ylabel('Count', fontsize=16)
plt.title('Education vs Loan status',fontsize=18)
plt.legend(Loan_Status,loc=1)
plt.show()
```




From the chart above, we can conclude** that the applicants who are graduate were in higher percentage of loan approval than non-graduate applicants.

In [12]:
```python
#Relationship between Self-Employed vs Loan_Status:
```

Home Page - Select or create a n | prediction_model - Jupyter Note

localhost:8892/notebooks/prediction_model.ipynb

Booking.com | Booking.com | LastPass password... | Express VPN | Gmail | YouTube | Maps | News | Translate | get_storage | Flutte... | Other favorites

jupyter prediction_model (unsaved changes)

File Edit View Insert Cell Kernel Help

In [12]:
```
#Relationship between Self-Employed vs Loan_Status:

Self_Employed=['Yes', 'No']
Loan_Status=['Yes', 'No']
pos=np.arange(len(Self_Employed))
bar_width=0.30
Loan_Status_Yes=[56,366]
Loan_Status_NO=[26,166]

plt.bar(pos,Loan_Status_Yes,bar_width,color='green',edgecolor='black')
plt.bar(pos+bar_width,Loan_Status_NO,bar_width,color='orange',edgecolor='black')
plt.xticks(pos, Self_Employed)
plt.xlabel('Self Employed', fontsize=16)
plt.ylabel('Count', fontsize=16)
plt.title('Self Employed vs Loan status',fontsize=18)
plt.legend(Loan_Status,loc=1)
plt.show()
```


From the chart above, we can see that non self employed applicant were more approved for loan that the self employed applicant

In [13]:
```
#Relationship between Property area and Loan status

Property_Area=['Rural','Semiurban','Urban']
```

Home Page - Select or create a n | prediction_model - Jupyter Note

localhost:8892/notebooks/prediction_model.ipynb

Booking.com | Booking.com | LastPass password... | Express VPN | Gmail | YouTube | Maps | News | Translate | get_storage | Flutte... | Other favorites

jupyter prediction_model (unsaved changes)

File Edit View Insert Cell Kernel Help

From the chart above, we can see that non self employed applicant were more approved for loan that the self employed applicant

In [13]:
```
#Relationship between Property area and Loan status

Property_Area=['Rural','Semiurban','Urban']
Loan_Status=['Yes', 'No']
pos=np.arange(len(Property_Area))
bar_width=0.35
Loan_Status_Yes=[110,179,133]
Loan_Status_NO=[69,54,69]

plt.bar(pos,Loan_Status_Yes,bar_width,color='green',edgecolor='black')
plt.bar(pos+bar_width,Loan_Status_NO,bar_width,color='orange',edgecolor='black')
plt.xticks(pos, Property_Area)
plt.xlabel('Property Area', fontsize=16)
plt.ylabel('Count', fontsize=16)
plt.title('Property Area vs Loan Status',fontsize=18)
plt.legend(Loan_Status,loc=1)
plt.show()
```


From the chart above results we can infer that, the higher percentage of loan approval is for semi-urban houses followed by urban and rural houses.

Home Page - Select or create a n | prediction_model - Jupyter Note

localhost:8892/notebooks/prediction_model.ipynb

Booking.com | Booking.com | LastPass password... | Express VPN | Gmail | YouTube | Maps | News | Translate | get_storage | Flutte... | Other favorites

jupyter prediction_model (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Not Trusted Python 3 (ipykernel)

Run Code

In [14]:
```
#Relationship between Credit History and Loan Status:

Credit_History=['Bad','Medium','Good']
Loan_Status=['Yes', 'No']
pos=np.arange(len(Credit_History))
bar_width=0.35
Loan_Status_Yes=[7,37,378]
Loan_Status_NO=[82,13,97]

plt.bar(pos,Loan_Status_Yes,bar_width,color='green',edgecolor='black')
plt.bar(pos+bar_width,Loan_Status_NO,bar_width,color='orange',edgecolor='black')
plt.xticks(pos, Credit_History)
plt.xlabel('Credit History', fontsize=16)
plt.ylabel('Count', fontsize=16)
plt.title('Credit History vs Loan Status',fontsize=18)
plt.legend(Loan_Status,loc=2)
plt.show()
```


From the chart above results we can infer that, the higher percentage of loan approval is for customers with good credit history followed bycustomers with medium credit history and then the bad

In [15]:
```
plt.figure(figsize=(10,8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm');
```

Home Page - Select or create a n | prediction_model - Jupyter Note

localhost:8892/notebooks/prediction_model.ipynb

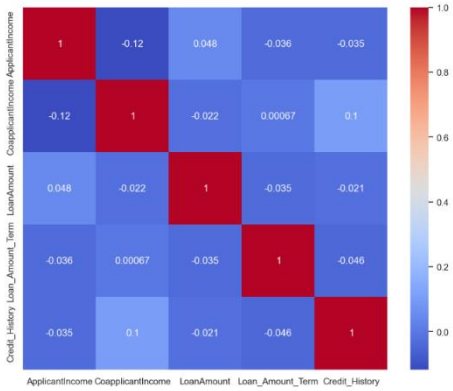
Booking.com | Booking.com | LastPass password... | Express VPN | Gmail | YouTube | Maps | News | Translate | get_storage | Flutte... | Other favorites

jupyter prediction_model (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Not Trusted Python 3 (ipykernel)

Run Code

In [15]:
```
plt.figure(figsize=(10,8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm');
```




In [16]:
```
# print out column with unique values
for col in data.columns:
    if data[col].dtypes == 'object':
        num_of_unique_cat = len (data[col].unique())
        print("Features '{col_name}' has '{unique_cat}' unique categories". format(col_name=col, unique_cat=num_of_unique_cat))
```

```
Features 'Loan_ID' has '614' unique categories
Features 'Gender' has '3' unique categories
Features 'Married' has '3' unique categories
```

Home Page - Select or create a n | prediction_model - Jupyter Note

localhost:8892/notebooks/prediction_model.ipynb

Booking.com | Booking.com | LastPass password... | Express VPN | Gmail | YouTube | Maps | News | Translate | get_storage | Flutte... | Other favorites

jupyter prediction_model (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help

Not Trusted | Python 3 (ipykernel)

Run | Code

In [16]:
```
# print out column with unique values
for col in data.columns:
    if data[col].dtypes == 'object':
        num_of_unique_cat = len (data[col].unique())
        print("Features '{col_name}' has '{unique_cat}' unique categories". format(col_name=col, unique_cat=num_of_unique_cat))
```

```
Features 'Loan_ID' has '614' unique categories
Features 'Gender' has '3' unique categories
Features 'Married' has '3' unique categories
Features 'Dependents' has '5' unique categories
Features 'Education' has '2' unique categories
Features 'Self_Employed' has '3' unique categories
Features 'Property_Area' has '3' unique categories
Features 'Loan_Status' has '2' unique categories
```

In [17]:
```
data = data.drop(['Loan_ID',], axis = 1)
```

In [18]:
```
data['Self_Employed'].replace({'Yes':1,'No':0},inplace=True)
data['Married'].replace({'Yes':1,'No':0},inplace=True)
data['Gender'].replace({'Male':1,'Female':0},inplace=True)
data['Education'].replace({'Graduate':1,'Not Graduate':0},inplace=True)
data['Property_Area'].replace({'Urban':2,'Semiurban':1,'Rural':0},inplace=True)
data['Loan_Status'].replace({'Y':1,'N':0},inplace=True)
```

In [19]:
```
y = data['Loan_Status'] #target
X = data.drop('Loan_Status', axis = 1) #predictors
```

In [20]:
```
#Standardize the data - Feature Scaling

from sklearn.preprocessing import StandardScaler
X['Dependents'] = X['Dependents'].replace('3+', 10)

sc_X = StandardScaler()
X_scaled = pd.DataFrame(sc_X.fit_transform(X), columns=X.columns)
```

In [21]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
```

In [22]:
```
model = LogisticRegression() #define the model
model.fit(X_train, y_train) #fit the model
ypred = model.predict(X_test) #predict on test sample
evaluation = f1_score(y_test, ypred)
evaluation
```

Out[22]:
```
0.8421052631578947
```

Home Page - Select or create a n | prediction_model - Jupyter Note

localhost:8892/notebooks/prediction_model.ipynb

Booking.com | Booking.com | LastPass password... | Express VPN | Gmail | YouTube | Maps | News | Translate | get_storage | Flutte... | Other favorites

jupyter prediction_model (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help

Not Trusted | Python 3 (ipykernel)

Run | Code

In [18]:
```
data['Self_Employed'].replace({'Yes':1,'No':0},inplace=True)
data['Married'].replace({'Yes':1,'No':0},inplace=True)
data['Gender'].replace({'Male':1,'Female':0},inplace=True)
data['Education'].replace({'Graduate':1,'Not Graduate':0},inplace=True)
data['Property_Area'].replace({'Urban':2,'Semiurban':1,'Rural':0},inplace=True)
data['Loan_Status'].replace({'Y':1,'N':0},inplace=True)
```

In [19]:
```
y = data['Loan_Status'] #target
X = data.drop('Loan_Status', axis = 1) #predictors
```

In [20]:
```
#Standardize the data - Feature Scaling

from sklearn.preprocessing import StandardScaler
X['Dependents'] = X['Dependents'].replace('3+', 10)

sc_X = StandardScaler()
X_scaled = pd.DataFrame(sc_X.fit_transform(X), columns=X.columns)
```

In [21]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
```

In [22]:
```
model = LogisticRegression() #define the model
model.fit(X_train, y_train) #fit the model
ypred = model.predict(X_test) #predict on test sample
evaluation = f1_score(y_test, ypred)
evaluation
```

Out[22]:
```
0.8421052631578947
```

In [23]:
```
pickle.dump(model,open('../Flask/model.pkl','wb'))
```

In [24]:
```
model = pickle.load(open('../Flask/model.pkl','rb'))
print(model)
```

```
LogisticRegression()
```

In [ ]:

Home.html

```html
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
    <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">

    <title>Predicting Personal Loan Approval Using Machine Learning</title>
  </head>
  <body>
<!-- This example requires Tailwind CSS v2.0+ -->
<div class="relative bg-white overflow-hidden">
  <div class="max-w-7xl mx-auto">
    <div class="relative z-10 pb-8 bg-white sm:pb-16 md:pb-20 lg:max-w-2xl lg:w-full lg:pb-28 xl:pb-32">
      <svg class="hidden lg:block absolute right-0 inset-y-0 h-full w-48 text-white transform translate-x-1/2" fill="currentColor"
        <polygon points="50,0 100,0 50,100 0,100" />
      </svg>

      <div class="relative pt-6 px-4 sm:px-6 lg:px-8">
        <nav class="relative flex items-center justify-between sm:h-10 lg:justify-start" aria-label="Global">
          <div class="flex items-center flex-grow flex-shrink-0 lg:flex-grow-0">
            <div class="flex items-center justify-between w-full md:w-auto">
              <a href="#">
                <span class="sr-only">Workflow</span>
                <img class="h-8 w-auto sm:h-10" src="https://tailwindui.com/img/logos/workflow-mark-indigo-600.svg">
              </a>
              <div class="-mr-2 flex items-center md:hidden">
```

```html
            To: "opacity-0 scale-95"
        -->
          <div class="absolute top-0 inset-x-0 p-2 transition transform origin-top-right md:hidden">
            <div class="rounded-lg shadow-md bg-white ring-1 ring-black ring-opacity-5 overflow-hidden">
              <div class="px-5 pt-4 flex items-center justify-between">
                <div>
                  <img class="h-8 w-auto" src="https://tailwindui.com/img/logos/workflow-mark-indigo-600.svg" alt="">
                </div>
                <div class="-mr-2">
                  <button type="button" class="bg-white rounded-md p-2 inline-flex items-center justify-center text-gray-400 hover:
                    <span class="sr-only">Close main menu</span>
                    <!-- Heroicon name: outline/x -->
                    <svg class="h-6 w-6" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke="currentColor" a
                      <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M6 18L18 6M6 6l12 12" />
                    </svg>
                  </button>
                </div>
              </div>
            <div class="px-2 pt-2 pb-3 space-y-1">
              <a href="#" class="block px-3 py-2 rounded-md text-base font-medium text-gray-700 hover:text-gray-900 hover:bg-gray-5

              <a href="#" class="block px-3 py-2 rounded-md text-base font-medium text-gray-700 hover:text-gray-900 hover:bg-gray-5

              <a href="#" class="block px-3 py-2 rounded-md text-base font-medium text-gray-700 hover:text-gray-900 hover:bg-gray-5

              <a href="#" class="block px-3 py-2 rounded-md text-base font-medium text-gray-700 hover:text-gray-900 hover:bg-gray-5
            </div>

          </div>
        </div>
```

```html
            <div class="-mr-2 flex items-center md:hidden">
              <button type="button" class="bg-white rounded-md p-2 inline-flex items-center justify-center text-gray-400 hover:t
                <span class="sr-only">Open main menu</span>
                <!-- Heroicon name: outline/menu -->
                <svg class="h-6 w-6" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke="currentColor" ar
                  <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M4 6h16M4 12h16M4 18h16" />
                </svg>
              </button>
            </div>
          </div>
        </div>
        <div class="hidden md:block md:ml-10 md:pr-4 md:space-x-8">
          <a href="#" class="font-medium text-gray-500 hover:text-gray-900">Home</a>

          <a href="#" class="font-medium text-gray-500 hover:text-gray-900">Prediction</a>

          <a href="#" class="font-medium text-gray-500 hover:text-gray-900">About us</a>

          <a href="#" class="font-medium text-gray-500 hover:text-gray-900">contact</a>

        </div>
      </nav>
    </div>

    <!--
      Mobile menu, show/hide based on menu open state.

      Entering: "duration-150 ease-out"
        From: "opacity-0 scale-95"
        To: "opacity-100 scale-100"
```

```html
    <main class="mt-10 mx-auto max-w-7xl px-4 sm:mt-12 sm:px-6 md:mt-16 lg:mt-20 lg:px-8 xl:mt-28">
      <div class="sm:text-center lg:text-left">
        <h1 class="text-4xl tracking-tight font-extrabold text-gray-900 sm:text-5xl md:text-6xl">
          <span class="block xl:inline">Predicting Personal Loan Approval Using Machine Learning</span>
        </h1>
          <h2>
              <span class="block text-indigo-600 xl:inline">Welcome to our Loan Approval Prediction Application</span>
          </h2>
        <p class="mt-3 text-base text-gray-500 sm:mt-5 sm:text-lg sm:max-w-xl sm:mx-auto md:mt-5 md:text-xl lg:mx-0">
          Our application is designed to help you predict whether or not your personal loan application will be approved by a ba
        </p>
        <div class="mt-5 sm:mt-8 sm:flex sm:justify-center lg:justify-start">
          <div class="rounded-md shadow">
            <a href="/predict" class="w-full flex items-center justify-center px-8 py-3 border border-transparent text-base font
             Predict Loan Approval
            </a>
          </div>

        </div>
      </div>
    </main>
  </div>
</div>
<div class="lg:absolute lg:inset-y-0 lg:right-0 lg:w-1/2">
  <img class="h-56 w-full object-cover sm:h-72 md:h-96 lg:w-full lg:h-full" src="https://images.unsplash.com/photo-1551434678-e0
</div>
</div>
```

```html
<p class="mt-5 text-base text-gray-500 sm:mt-5 sm:text-lg sm:max-w-xl sm:mx-auto md:mt-5 md:text-xl lg:mx-0">
  Our application is designed to help you predict whether or not your personal loan application will be approved by a ba
</p>
<div class="mt-5 sm:mt-8 sm:flex sm:justify-center lg:justify-start">
  <div class="rounded-md shadow">
    <a href="/predict" class="w-full flex items-center justify-center px-8 py-3 border border-transparent text-base font
     Predict Loan Approval
    </a>
  </div>

</div>
</div>
</main>
</div>
</div>
<div class="lg:absolute lg:inset-y-0 lg:right-0 lg:w-1/2">
<img class="h-56 w-full object-cover sm:h-72 md:h-96 lg:w-full lg:h-full" src="https://images.unsplash.com/photo-1551434678-e6
</div>
</div>



<!-- Option 1: Bootstrap Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-JEW9xMcG8R-
</body>
</html>
```

Input.html

```html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/html">
<head>

<style>
/* Style inputs with type="text", select elements and textareas */
input[type=number], select, textarea {
  width: 100%; /* Full width */
  padding: 12px; /* Some padding */
  border: 1px solid #ccc; /* Gray border */
  border-radius: 4px; /* Rounded borders */
  box-sizing: border-box; /* Make sure that padding and width stays in place */
  margin-top: 6px; /* Add a top margin */
  margin-bottom: 16px; /* Bottom margin */
  resize: vertical /* Allow the user to vertically resize the textarea (not horizontally) */
}

/* Style the submit button with a specific background color etc */
input[type=submit] {
  background-color: #04AA6D;
  color: white;
  padding: 12px 20px;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

/* When moving the mouse over the submit button, add a darker green color */
input[type=submit]:hover {
  background-color: #45a049;
```

```html
    /* Add a background color and some padding around the form */
    .container {
      border-radius: 5px;
      background-color: #f2f2f2;
      padding: 20px;
    }

    body {
      background-size: cover;
      background-image: url('https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcR70TDTAUcRk7Q7W2nK-aIqsmoTN6VruMi0mA&usqp=CAU')
    }
    </style>
</head>
<body>

<h3>Enter your Details for Loan Approval Prediction</h3>

<div class="container">

    <form action = '/submit', method = 'post'>

    <label for="Gender">Gender</label>
    <select id="Gender" name="Gender">
      <option value=0>Male</option>
      <option value=1>Female</option>
    </select>

    <label for="Married">Married</label>
    <select id="Married" name="Married">
      <option value=1>Yes</option>
```

```html
    <label for="Married">Married</label>
    <select id="Married" name="Married">
      <option value=1>Yes</option>
      <option value=0>No</option>
    </select>



    <label for="Dependents">Dependents</label>
    <input type="number" id="Dependents" min = 0 max = 10 name="Dependents" placeholder="No of Dependents on you.....">



    <label for="Education">Education</label>
    <select id="Education" name="Education">
      <option value=1>Graduate</option>
      <option value=0>Not Graduate </option>
    </select>

     <label for="Self_Employed">Self Employed</label>
    <select id="Self_Employed" name="Self_Employed">
      <option value=1>Yes</option>
      <option value=0>No</option>
    </select>

    <label for="ApplicantIncome">Applicant Income</label>
    <input type="Number" min = 1000 id="ApplicantIncome" name="Applicant Income" placeholder="Your Income...">

    <label for="CoapplicantIncome">CO Applicant Income</label>
    <input type="Number" min = 100 id="CoapplicantIncome" name="Co Applicant Income" placeholder="Your Co Applicant Income...">
```

```html
<label for="ApplicantIncome">Applicant Income</label>
<input type="Number" min = 1000 id="ApplicantIncome" name="Applicant Income" placeholder="Your Income...">

<label for="CoapplicantIncome">CO Applicant Income</label>
<input type="Number" min = 100 id="CoapplicantIncome" name="Co Applicant Income" placeholder="Your Co Applicant Income...">

<label for="LoanAmount">Loan Amount</label>
<input type="Number" min = 0 id="LoanAmount" name="Loan Amount" placeholder="Enter the Loan Amount ...">

<label for="Loan_Amount_Term">Loan Amount Term</label>
<input type="Number" min = 30 max = 15000 id="Loan_Amount_Term" name="Loan Amount Term" placeholder="Enter the Term Loan Amou

<label for="Credit_History">Credit History</label>
<input type="Number" min = 0 max = 5 id="Credit_History" name="Credit History" placeholder="Enter the Your Previous Credit Hi

<label for="Property_Area">Property Area</label>
<select id="Property_Area" name="Property Area">
  <option value=2>Urban</option>
  <option value=0>Rural</option>
  <option value=1>Semi Urban</option>
</select>

<input type ='submit' value = 'Submit'>
<

</div>

</body>
</html>
```

Output.html

```html
<!DOCTYPE html>
<html>
<head>
<title>Home</title>
<style>
body
{
    background-image: url("https://www.dbs.com/in/iwov-resources/media/images/learn/banners/what-is-a-personal-loan-1404x630.jpg")
    background-size: cover;
}
.pd{
padding-bottom:45%;}
}
</style>
</head>
<body>
<form action="/submit" method="post">
<br>
<center><b class="pd"><font color="black" size="15" font-family="Comic Sans MS" >Loan Approval Prediction</font></b></center><br>
<div>
<br>
<center>
<h2><font color="black"> {{result}} </h2>
</center>
</div>
</form>
</body>
</html>
```

App.py

```python
import numpy as np
import pickle
import pandas
import os
from flask import Flask, request, render_template

# Create a Flask web application instance
app = Flask(__name__, template_folder='template')

# Load the trained model from a saved pickle file
model = pickle.load(open(r'model.pkl', 'rb'))

# Define a route to render the home page HTML template
@app.route('/')
def home():
    return render_template('home.html')

# Define a route to render the input HTML form
@app.route('/predict', methods=["POST","GET"])
def predict():
    return render_template("input.html")

# Define a route to handle form submission and display the prediction result
@app.route('/submit', methods=["POST","GET"])
def submit():
    # Read the input values submitted by the user
    input_feature = [int(x) for x in request.form.values()]
    # Convert the input values to a NumPy array
    input_feature = [np.array(input_feature)]
    # Define the column names for the input data frame
    names = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'ApplicantIncome'
```

```python
    return render_template("input.html")

# Define a route to handle form submission and display the prediction result
@app.route('/submit', methods=["POST","GET"])
def submit():
    # Read the input values submitted by the user
    input_feature = [int(x) for x in request.form.values()]
    # Convert the input values to a NumPy array
    input_feature = [np.array(input_feature)]
    # Define the column names for the input data frame
    names = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'ApplicantIncome',
        'CoapplicantIncome','LoanAmount','Loan_Amount_Term','Credit_History','Property_Area']
    # Create a Pandas data frame from the input values with the column names
    data = pandas.DataFrame(input_feature, columns=names)

    # Use the loaded model to make a prediction
    prediction = model.predict(data)
    # Convert the prediction from a NumPy array to an integer
    prediction = int(prediction)

    # Render the output HTML template with the prediction result
    if prediction == 0:
        return render_template("output.html",result="Loan will not be approved")
    else:
        return render_template("output.html",result="Loan will be approved")

# Start the Flask web application on port 8000
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8000, debug=True)
```